Sample-efficient Learning and Generalization with Text Representations

by

Lajanugen Logeswaran

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2021

Doctoral Committee:

Associate Professor Honglak Lee, Chair
Professor Qiaozhu Mei
Professor Rada Mihalcea
Professor Dragomir Radev

Lajanugen Logeswaran

llajan@umich.edu

ORCID iD: 0000-0002-1665-475X

ACKNOWLEDGEMENTS

I will start by thanking my advisor, Professor Honglak Lee. You took a chance on me 6 years ago when you hired me as a PhD student, and I couldn't be happier about my decision to come to Michigan and work with you. I have learned a lot from you about doing good research over the course of my PhD. Thank you for creating opportunities for me to grow professionally. I am also extremely thankful for the academic freedom to work on topics that I was passionate about. And most of all, thank you for supporting me through the many ups and downs in my graduate school life.

I wish to thank my industry collaborators with whom I had a chance to work with during my internships. I thank Samy Bengio, Kristina Toutanova, Ming-Wei Chang, Kenton Lee, Jacob Devlin, Marc'Aurelio Ranzato and Arthur Szlam for their mentoring and advice as well as their influence on my approach to research.

My labmates and visiting researchers in my group were a big part of my PhD journey. Thank you for inspiring me to always push harder. I learned a lot from all of you and I will cherish the friendships I have formed with you. I thank Scott Reed, Ruben Villegas, Junhyuk Oh, Xinchen Yan, Yuting Zhang, Seunghoon Hong, Kibok Lee, Jongwook Choi, Yunseok Jang, Sungryull Sohn, Wilka Carvalho, Yijie Guo, Kimin Lee and Anthony Liu.

I want to thank my committee members Dragomir Radev, Rada Mihalcea, Qiaozhu Mei and Honglak Lee for helping me with my thesis. Thank you for your valuable feedback and advice and for helping me in various other ways during my PhD.

Finally, I thank my family and friends for their incredible support. I would not be where I am today if not for my parents, you have always enabled me to do the things I am passionate about. I thank my wife for putting up with me all these years - It must not have been easy for you when I worked long hours, and the long-distance relationship on top of that. Your love and continued support has enabled me to reach this milestone.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

Appendix

ABSTRACT

Humans have a remarkable ability to learn without much supervision. Often, a few labelled instances or a single demonstration is enough for us to learn a new concept. Most of our knowledge is acquired in a weakly unsupervised manner, via reading, perception, and active interaction with the world. Machine learning models, on the other hand, struggle to learn from limited supervision and often need large amounts of labelled data to learn. In many practical instances, however, such supervision is not available. Furthermore, collecting labeled instances for training may be expensive or infeasible due to privacy reasons. This calls for approaches that can adapt to new tasks or new domains without needing a lot of labelled data.

In this thesis, I address the limited supervision problem from two perspectives. First, I examine methods that exploit large amounts of unlabelled data to learn useful feature representations in a self-supervised manner. Such representations capture rich prior knowledge about the data, allowing them to be useful across many tasks, and enable data-efficient learning of new tasks. In particular, my work is concerned with the following key questions pertaining to text representations - (i) How do we learn representations of larger units of text, beyond words? (ii) How do we design training objectives that can efficiently learn such representations? (iii) How do we come up with representations that allow efficient knowledge transfer to downstream language understanding tasks?

Second, I explore models and algorithms capable of learning from limited supervision. My work studies weakly supervised, few-shot and zero-shot learning settings with applications to text generation, sequence modeling, entity understanding and embodied control. My work demonstrates that text descriptions are an effective means of building models that generalize to new domains and new tasks without needing to experience supervised data for the new domain/task. I believe that the next generation of AI technologies will be driven by models that read and understand text to perform tasks.

CHAPTER I

Introduction

Language understanding is one of the hallmarks of human intelligence. Natural language is a fundamental means of human communication and has played an important role in the development of the human race. It is a rich and complex signal, and yet we are able to comprehend and use language in our day to day activities almost effortlessly, an ability we struggle to replicate in our machines.

Machine comprehension of human languages has become an increasingly critical need in our times. Machine translation systems such as Google Translate, conversational systems such as Alexa and Google Home and text auto-complete features such as Smart Reply are a few of the tools we use on a daily basis which are powered by AI. Despite the impressive progress made on such problems, modern machine learning methods have not achieved a level of maturity where a system could conduct a coherent conversation with a human.

Deep Neural Networks have given rise to many of the systems described previously. The Deep Learning revolution began early this decade, driven by the growing amount of compute and data. Advances in model architectures and optimization algorithms have acted as building blocks in many of these successes. But these advances have largely relied on the availability of large labelled datasets. Such datasets are often not available in many practical scenarios.

Consider an example where we want to build an entity recognition system for legal documents. If we examine how traditional Entity Linking systems are built, they tend to rely quite heavily on structured annotated resources. However, such resources are not readily available for practical domains of interest such as legal documents or internal documents of a company. As a result, models and algorithms designed for the scenario where such resources are available fail to be applicable to these practical settings. This calls for methods that are capable of generalizing to new settings with limited supervision.

My work tackles the limited supervision problem from two perspectives. First, I examine

methods for using large scale unlabelled text to learn useful text representations. Such general purpose representations improve the sample efficiency of text understanding methods to learn new tasks. Second, I propose models and algorithms that are able to learn tasks and generalize to new settings with limited supervision.

1.1 Motivation

1.1.1 Unsupervised learning of text representations

Unsupervised text representations have had a transformative impact on Natural Language Processing. Training representations of words (Mikolov et al., 2013), sentences (Kiros et al., 2015), paragraphs (Le and Mikolov, 2014) and entire spans of text (Devlin et al., 2019) have driven progress across many tasks in NLP. Training these representations in an unsupervised manner is especially attractive due to the availability of unlabelled data in abundance. It is thus imperative to explore methods that enable us to train data representations from large scale unlabelled corpora in an effective manner.

Prior work on learning text representations typically use reconstruction based training objectives. Prominent types of training objectives include context prediction (Kiros et al., 2015), de-noising (Hill et al., 2015) and language modeling (Devlin et al., 2019). While such reconstruction based training objectives have been quite successful, they have some limitations. Reconstructing words can be computationally expensive due to auto-regressive decoding or large output softmaxes when using large word vocabularies. A more subtle issue with reconstruction is that model capacity is spent on learning to predict unimportant aspects of text that do not contribute to it's semantics.

Approach	Training Dataset	Training Time
SkipThoughts	Books corpus (4G)	9 days (GTX 1080)
BERT	Books corpus + Wikipedia (17G)	4 days (4-16 Cloud TPUs)
QuickThoughts	Books corpus (4G)	12 hrs (1 TitanX GPU)
QuickThoughts	Books corpus + UMBC corpus (12G)	1 day (1 TitanX GPU)

Table 1.1: Training time of popular representation learning approaches in the literature (SkipThoughts and BERT) and our proposed QuickThoughts approach (chapter III).

Table 1.1 shows the training time for two popular representation learning approaches on commonly used unlabelled corpora. We see that these methods can be fairly expensive to train, sometimes requiring specialized hardware such as TPUs. The size of these unlabelled corpora are also modest by today's standards. Unlabelled data is available in large quantities and recent work have used as much as 170G of unlabelled data for training models (Liu et al., 2019). Exploring scalable alternatives to these methods is imperative for the following

reasons. First, representations generally improve with more data, and we need efficient methods to be able to exploit such data. Second, to explore novel training objectives, we need to be able to iterate efficiently. Waiting for several days to assess the representations makes these iterations expensive and prohibits new advances. And finally, the significant carbon footprint and environmental impact of training these models cannot be ignored.

As a departure from prior work, my work considers self-supervised classification training objectives (chapters III, IV). I consider discourse signals such as sentence order information available in large unlabelled corpora to train representations. In addition to producing useful representations, such methods are also computationally more efficient to train compared to their generative counterparts and lead to models that train in under a day. One of my key contributions is the development of contrastive learning methods to train high quality sentence representations efficiently from large scale unlabelled text.

1.1.2 Models and algorithms to learn from limited supervision

While unsupervised text representations have enabled progress in many tasks, they typically assume that adequate labelled data exists to learn a particular target task. In many practical scenarios such labels are scarce and it is important to think about how these representation models can be exploited in data-starved settings.

In this thesis, I approach learning in data-starved settings from two perspectives. First, I explore training algorithms for adapting the transformer architecture, which has been widely successful for transfer learning using pre-trained text representations, to the few-shot learning setting. Second, I pursue a learning paradigm where models learn to adapt to new tasks or new domains by exploiting pre-trained text representations.

Few-shot learning with transformers Transformers (Vaswani et al., 2017a) have been very successful at modeling discrete sequences (Barrault et al., 2019; Devlin et al., 2019; Parisotto et al., 2019). A strong prerequisite for learning in these settings is the availability of large labelled datasets, which is often not available in practice. I explore the applicability of the transformer architecture in the limited data setting.

In this thesis, I am interested in two interesting questions pertinent to transformer models in the limited data regime.

- How do we adapt the transformer architecture to the few-shot regime?
- How do we exploit the text understanding capabilities of pre-trained transformer models to perform few-shot learning?

I propose an optimization based meta-learning algorithm to address the first question where transformer models learn a new task by predicting an appropriate *task embedding* vector

(chapter V). In relation to the second question, I exploit the reading comprehension abilities of pre-trained transformer language models to perform zero-shot learning (chapter VI).

Zero-shot generalization with text descriptions While the previous two research thrusts explore representation learning and learning from limited supervision, here we examine how strong pre-trained representations can help build models that can learn and generalize better with limited supervision. I focus on a learning paradigm where models rely on text descriptions to learn and generalize, drawing inspiration from how humans are able to learn to perform new tasks by reading instructions in the form of text. For instance, we can cook a meal just by reading a recipe or assemble furniture by reading an instruction manual. Even if we have no prior experience with cooking or assembling furniture, we still manage to complete these tasks.

Recent advances in text understanding provide strong motivation for exploiting this type of *natural language supervision*. Pre-trained transformer models have boosted the performance on language understanding problems such as reading comprehension in the recent years. Remarkably, models like GPT-3 exhibit few-shot generalization when provided with a natural language prompt of a task. This shows that we now have models that can read and understand text to a much greater extent than we were able to until few years ago. This provides a compelling reason to think about building AI that acquires better generalization capabilities by reading text.

One of the key questions I try to answer in this thesis is, how can we use text and text representations to expand the generalization abilities of AI? Specifically, I demonstrate how generalization capabilities of entity linking systems can be improved by framing it as a reading comprehension task and exploiting strong models from the reading comprehension literature. I also present embodied agents that learn from text instructions and generalize to unseen tasks by making use of task descriptions.

1.2 Thesis Statement

This thesis establishes that the *sample efficiency and generalization properties of machine learning models can be improved with text representations*.
Towards this end, I make the following contributions in this thesis

- Efficient discriminative self-supervised training objectives to learn representations from large scale unlabelled text. I present QuickThought vectors in chapter III and representations based on text order reconstruction in chapter IV.

- Models and algorithms capable of learning sequence tasks from limited supervision (chapter V).

- Models that generalize to new domains (chapter VI) and tasks (chapter VII) in a zero-shot manner by exploiting text representations.

1.3 Impact of thesis

My work on sentence representations has led to a renewed interest in learning text representations based on contrastive training objectives (Rethmeier and Augenstein, 2021). It has served as an inspiration for incorporating discriminative training objectives in modern pre-training methods such as BERT (Devlin et al., 2019). My work on zero-shot entity linking is the first of its kind to study the entity linking problem in a zero-shot setting by posing it as a reading comprehension task. This work received a nomination for best paper at ACL 2021. My work has inspired subsequent work such as BLINK (Wu et al., 2019) to apply pre-trained transformer methods to recognize unseen entities in Wikipedia. In addition, I introduced a multi-stage pre-training approach for unsupervised domain adaptation and it was shown by subsequent influential work (Gururangan et al., 2019) that the same approach is effective for other tasks and data domains.

1.4 Thesis Outline

My main contributions are presented in chapters III-VII and I conclude with future directions in chapter VIII. A brief overview of the chapters and the respective publications discussed is given below.

In chapters III, IV I explore self-supervised training objectives based on discourse signals to learn general purpose text representations. I show that learning to recover the order of sentences that appear in a document and learning to predict the next sentence given the current sentence from a document leads to useful representations. In addition to producing high quality representations, I show that this leads to more efficient training and scales better to large datasets than prior methods.

- Lajanugen Logeswaran, Honglak Lee. An efficient framework for learning sentence representations. *ICLR 2018*.

- Lajanugen Logeswaran, Honglak Lee, Dragomir Radev. Sentence Ordering and Coherence Modeling using Recurrent Neural Networks. *AAAI 2018*.

In chapters V, VI, VII I explore how text representations can enable machine learning models that generalize to new domains and tasks without needing to experience a lot of supervision. Chapter V examines sequence classification and transduction problems in the few shot setting, where I present transformer models that can be adapted to new tasks

by inferring an appropriate *task embedding* vector. Chapter VI describes an approach to entity linking where I assume that text descriptions of entities are available and show that pre-trained representation models help generalize to unseen entities in unseen domains in a zero-shot manner. Chapter VII presents embodied agents that generalize to unseen tasks by exploiting the compositional nature of task descriptions.

- Lajanugen Logeswaran, Ann Lee, Myle Ott, Honglak Lee, Marc'Aurelio Ranzato, Arthur Szlam. Few-shot Sequence Learning with Transformers. *NeurIPS Workshop on Meta-Learning (MetaLearn 2020)*.
- Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, Honglak Lee. Zero-Shot Entity Linking by Reading Entity Descriptions. *ACL 2019*.
- Lajanugen Logeswaran, Wilka Carvalho, Honglak Lee. Learning Compositional Tasks from Text Instructions. ***In submission***.

CHAPTER II

Background

This chapter discusses background and preliminaries for subsequent chapters. Section 2.1 presents background on text representation learning. In section 2.2 I will discuss approaches to learning from limited supervision including meta-learning, few-shot learning and zero-shot learning.

2.1 Representation Learning

Pre-deep learning AI systems were dominated by feature-based approaches. These features were generally handcrafted for each task. While engineering these features is a hard task on its own, such features have restricted applicability. Features designed for one particular problem domain may be sub-optimal for another. It is more favorable to let the learning algorithm figure out these features on its own. This eliminates human effort and more importantly, learned features have the potential to apply across many different problems.

Much of the progress made in language and vision, and perhaps other data modalities, can be attributable to models and algorithms that learn good feature representations from raw data. The Word2vec (Mikolov et al., 2013) algorithm is a fundamental method for learning representations of words, which laid the foundations for modern deep learning based NLP. Similarly, in the vision domain, CNNs trained on the ImageNet classification dataset have been demonstrated to be useful for many vision tasks (Krizhevsky et al., 2012).

We categorize text representation learning approaches along the following dimensions based on how the representations are trained and how the representations are used in other tasks. Table 2.1 provides an overview of prior work along these dimensions of categorization.

2.1.1 Representation learning approaches

Both supervised and unsupervised approaches have been studied for training text representations. We review these approaches in this section.

Supervised	SBERT (Reimers and Gurevych, 2019) USE (Cer et al., 2018) GenSen (Subramanian et al., 2018) InferSent (Conneau et al., 2017) CoVe (McCann et al., 2017)	MT-DNN (Liu et al., 2019)
Unsupervised	*QuickThought (Logeswaran and Lee, 2018)* *Sentence Ordering (Logeswaran et al., 2018)* SDAE (Hill et al., 2016) SkipThoughts (Kiros et al., 2015) ParagraphVec (Le and Mikolov, 2014)	T5 (Raffel et al., 2019) GPT (Radford et al., 2019) BERT (Devlin et al., 2019) ELMo (Peters et al., 2018b) SA-LSTM (Dai and Le, 2015)
	Text encoder	**Pre-trained model**

Table 2.1: (Non-exhaustive) Summary of representation learning approaches categorized based on whether they use labelled data for training (supervised/unsupervised) and how the trained representations are used in downstream tasks (as feature representations/model initialization).

Supervised representation learning Prior work have trained representations from human-annotated data. InferSent (Conneau et al., 2017) and CoVe (McCann et al., 2017) are popular approaches which respectively use datasets for Natural Language Inference (NLI) and Machine Translation (MT) to train representations. Multitask training on both labelled and unlabelled text has been used in GenSen (Subramanian et al., 2018) and Universal Sentence Encoder (USE) (Cer et al., 2018) where models are trained on a combination of supervised and unsupervised training losses. Yet another class of approaches take a model that has been pre-trained using unlabelled text and further fine-tune them on labelled data. SBERT (Reimers and Gurevych, 2019) takes a pre-trained BERT model and fine-tunes it using NLI data. MT-DNN (Liu et al., 2019) fine-tunes a BERT model on annotated data from multiple tasks.

Unsupervised representation learning Unsupervised representation learning refers to learning representations solely based on unlabelled data, which are usually available in abundance. A particular variant of unsupervised representation learning called *self-supervised learning* has recently become popular, where the idea is to use labels that are naturally available as part of the data for supervision. For instance, language modeling is a self-supervised task where models learn to predict a word given its context. The label to be predicted in this case is a word, which occurs naturally in the data, and doesn't require any external supervision. Being able to exploit abundant unlabelled data is an attractive property of these methods.

A range of self-supervised training objectives have appeared in the literature. Many

early representation learning work were based on the distributional hypothesis - the idea that the meaning of words, phrases or sentences is determined by the context in which they appear. Such methods assign a vector representation to units of text such that text that appears in similar contexts have similar embedding vectors. Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and SkipThought vectors (Kiros et al., 2015) are prominent examples of work based on the distributional hypothesis.

Another prominent class of algorithms based on text reconstruction have received more attention in the recent years. These methods can be broadly classified under language modeling and de-noising approaches. De-noising approaches introduce noise to a given piece of text and reconstruct the original text, typically with an encoder-decoder model (Glorot et al., 2011; Hill et al., 2015).

ELMo (Peters et al., 2018a) ignited the popularity of language modeling as a self-supervised task for training representations. With the advent of the transformer architecture (Vaswani et al., 2017b), language modeling on unlabelled text with a transformer (Radford et al., 2018) proved to be much more successful than with other sequence models like RNNs. Scaling up these methods using bigger models and large unlabelled corpora continues to produce models and representations highly useful for language understanding (Brown et al., 2020). A limitation of language modeling that makes it suboptimal for text understanding applications is the lack of bi-directional context modeling.

BERT (Devlin et al., 2019) considers a hybrid training objective called masked language modeling inspired by language modeling and de-noising. It's ability to model bi-directional context resulted in strong performance across NLP benchmarks and spurred a significant interested in transformer models trained as masked language models (Lewis et al., 2019; Joshi et al., 2020; Song et al., 2019).

2.1.2 Representation usage in downstream tasks

We can also categorize text representation models based on how they are used in downstream tasks. We review these approaches below.

Representations as features vectors A straightforward way in which trained representations can be used is as feature representations in downstream tasks. Models that produce such feature representations are called *text encoders*. Text encoders attempt to produce a concise fixed-length representation of a given span of text. We can then apply traditional feature based Machine Learning algorithms and train models with supervised learning. Supervised and unsupervised approaches to train text encoders have been considered in prior work (See first column of table 2.1).

Text encoders are particularly useful in applications where supervision is limited or

computational efficiency is a priority. Training a classifier with few parameters on top of feature vectors is an effective way to learn tasks with few labelled data (Kiros et al., 2015). Large scale retrieval applications is another instance where text encoders are highly beneficial. Query and candidate texts can be embedded in feature space and can be efficiently compared using inner product or cosine distance (Reimers and Gurevych, 2019).

Representation learning as pre-training Another paradigm of transfer learning that has enjoyed a lot of success is *pre-training*. Compared to training a text encoder to extract feature representations, the text encoder itself can be further fine-tuned to adapt representations to the target task. In addition to the text encoder, task specific architectures and parameters can be introduced and the overall model trained end-to-end for the given task. Training the text encoder can be thought of as finding a good initialization for this part of the model (hence the term *pre-training*). Pre-training models with unsupervised learning has become the standard approach to NLP due to the significant benefits offered by unsupervised pre-training (See second column of table 2.1). In contrast to the text encoder approach, pre-training tends to be much more successful when sufficient labelled data exists for the target task to fine-tune the model.

2.2 Learning from limited supervision

The advances in deep learning have been accompanied by the need for large, carefully curated, labelled datasets. This limits the applicability of these models in practical situations. In many cases, collecting labelled data can be infeasible due to cost, security or privacy reasons. On the other hand, models which rely on annotated resources have limited generalization properties. If the task, the data domain or the label space changes, the model is either no longer applicable or needs to be re-trained using data from the new setup. This raises two important questions about machine learning models in practical applications.

- How do we get models to learn in settings where labelled data is scarce?
- How do we improve the generalization properties of machine learning models so that they are applicable outside the tasks or domains for which they are trained?

Limited supervision The limited supervision problem manifests in different ways in practice. In the simplest setting, we might have a few labels to learn a desired task. This problem is typically addressed in the semi-supervised, meta-learning and few-shot learning literature. In many cases, we may not have supervision for the target task, but we might have supervision for a different, but related task. In such cases, weakly supervised methods are applicable. Finally, we might have supervision for the desired task, but in a data domain

Similarity based
- Prototypical Networks (Snell et al., 2017)
- Matching Networks (Vinyals et al., 2015a)

Memory based
- GPT-3 (Brown et al., 2020)
- SNAIL (Mishra et al., 2018)
- MANN (Santoro et al., 2016)

Optimization based
- *TAM (Logeswaran and Lee, 2018)*
- CAVIA (Zintgraf et al., 2019)
- MAML (Finn et al., 2017)

Table 2.2: Overview of meta-learning and few-shot learning approaches.

Attribute information
- ESZSL (Romera-Paredes and Torr, 2015)
- SJE (Akata et al., 2015)

Word embedding
- Socher et al. (2013a)
- CoSE (Norouzi et al., 2013)

Text descriptions
- *Zeshel (Logeswaran et al., 2019)*
- Elhoseiny et al. (2013)
- Branavan et al. (2012)

Table 2.3: Overview of zero-shot learning approaches.

different from the domain of interest. Depending on the problem, this can be a domain adaptation and/or zero-shot learning problem. We briefly review some of these learning paradigms relevant to this thesis below.

Meta Learning In the traditional training paradigm, models learn a task from a set of training instances and are expected to perform well on unseen instances from the distribution of training instances. In contrast, meta learning or learning-to-learn (Schmidhuber, 1987; Hochreiter et al., 2001; Andrychowicz et al., 2016; Finn et al., 2017; Nichol and Schulman, 2018) seeks to learn *how to learn a new task* given data for that task. In this setup, we are given a set of training tasks, each of which are composed of labelled instances. Models are expected to do well on instances of an unseen task at test time.

Meta-gradient approaches to meta learning are popular in the deep-learning literature. In meta-learning, we seek to determine a set of parameters for which optimizing a model based on the training loss for a new task (typically with a small number of training instances) produces a model that performs well on heldout instances of the same task. Meta-gradient approaches operationalize precisely this notion with a bi-level optimization problem where the inner loop corresponds to updating model parameters on training instances and the outer loop is defined based on the performance of the updated parameters on unseen instances.

Prior approaches differ based on what parameters are being optimized in the outer loop. MAML (Finn et al., 2017) consider the model parameters to be the parameters being optimized in the outer loop. This essentially means that we are trying to find a model initialization which can be easily adapted to a new task using a few parameter updates on training instances of that task. Other variants such as CAVIA (Zintgraf et al., 2019) consider a subset of model parameters for the outer loop. Similarly Ravi and Larochelle

(2016) perform outer loop updates on parameters of an optimizer, i.e., we learn an optimizer capable of effectively learning new tasks.

Few-shot Learning Few-shot learning refers to learning a task from a handful of training examples. In a k-shot learning setup, k represents the number of training examples available to learn a task at test time, where k is typically less than 20. Few-shot learning is closely related to meta learning and these two terms are often used interchangeably. Meta learning approaches are often evaluated on few-shot problems since meta-learned skills are most useful when the new task has limited supervision.

Few-shot learning has received a lot of interest in the computer vision domain. They can broadly be classified under similarity based methods, memory based methods and optimization based approaches (See table 2.2 for an overview). Similarity based methods are applicable to classification problems and rely on training a similarity function between instance pairs (Vinyals et al., 2016; Snell et al., 2017). A query instance can then be classified based on it's similarity to instances belonging to a particular class label. Memory based models construct a memory consisting of training instances of a task and learn to exploit the memory to perform prediction on query instances of that task (Santoro et al., 2016; Mishra et al., 2018; Brown et al., 2020).

Adapting these methods to the NLP literature can be non trivial due to the sequential nature of language and the class of problems we care about in NLP. In this thesis we present approaches to adapt transformer models that have been quite successful in NLP problems to the few-shot setting.

Zero-shot Learning An extreme version of few-shot learning where no labelled examples are available for a test task is called zero-shot learning. To make learning feasible in this setup, it is typically assumed that some additional metadata is available for the target task of interest (See table 2.3 for an overview of prior work). For example, in a classification setting where the model is required to generalize to unseen class labels, metadata such as attribute information (Romera-Paredes and Torr, 2015; Akata et al., 2015) or word descriptions (Socher et al., 2013a; Norouzi et al., 2013) of the class labels are assumed to be available.

In particular, we focus on how to exploit rich text information to drive zero-shot generalization to new domains and tasks. Early work have explored building visual classifiers from text descriptions (Elhoseiny et al., 2013) and learning to play games from text manuals (Branavan et al., 2012). More recently, large pre-trained language models have been used to exploit text descriptions for zero-shot learning (Radford et al., 2021).

CHAPTER III

An Efficient Framework for Learning Sentence Representations

This chapter presents a simple and efficient framework for learning sentence representations from unlabelled data. Drawing inspiration from the distributional hypothesis and recent work on learning sentence representations, we reformulate the problem of predicting the context in which a sentence appears as a classification problem. Given a sentence and its context, a classifier distinguishes context sentences from other contrastive sentences based on their vector representations. This allows us to efficiently learn different types of encoding functions, and we show that the model learns high-quality sentence representations. We demonstrate that our sentence representations outperform state-of-the-art unsupervised and supervised representation learning methods on several downstream NLP tasks that involve understanding sentence semantics while achieving an order of magnitude speedup in training time.

3.1 Introduction

Methods for learning meaningful representations of data have received widespread attention in recent years. It has become common practice to exploit these representations trained on large corpora for downstream tasks since they capture a lot of prior knowlege about the domain of interest and lead to improved performance. This is especially attractive in a transfer learning setting where only a small amount of labelled data is available for supervision.

Unsupervised learning allows us to learn useful representations from large unlabelled corpora. The idea of self-supervision has recently become popular where representations are learned by designing learning objectives that exploit labels that are freely available with the data. Tasks such as predicting the relative spatial location of nearby image patches (Doersch

et al., 2015), inpainting (Pathak et al., 2016) and solving image jigsaw puzzles (Noroozi and Favaro, 2016) have been successfully used for learning visual feature representations. In the language domain, the distributional hypothesis has been integral in the development of learning methods for obtaining semantic vector representations of words (Mikolov et al., 2013). This is the assumption that the meaning of a word is characterized by the word-contexts in which it appears. Neural approaches based on this assumption have been successful at learning high quality representations from large text corpora.

Recent methods have applied similar ideas for learning sentence representations (Kiros et al., 2015; Hill et al., 2016; Gan et al., 2016). These are encoder-decoder models that learn to predict/re-construct the context sentences of a given sentence. Despite their success, several modelling issues exist in these methods. There are numerous ways of expressing an idea in the form of a sentence. The ideal semantic representation is insensitive to the form in which meaning is expressed. Existing models are trained to reconstruct the surface form of a sentence, which forces the model to not only predict its semantics, but aspects that are irrelevant to the meaning of the sentence as well.

The other problem associated with these models is computational cost. These methods have a word level reconstruction objective that involves sequentially decoding the words of target sentences. Training with an output softmax layer over the entire vocabulary is a significant source of slowdown in the training process. This further limits the size of the vocabulary and the model (Variations of the softmax layer such as hierarchical softmax (Mnih and Hinton, 2009), sampling based softmax (Jean et al., 2014) and sub-word representations (Sennrich et al., 2015) can help alleviate this issue).

We circumvent these problems by proposing an objective that operates directly in the space of sentence embeddings. The generation objective is replaced by a discriminative approximation where the model attempts to identify the embedding of a correct target sentence given a set of sentence candidates. In this context, we interpret the 'meaning' of a sentence as the information in a sentence that allows it to predict and be predictable from the information in context sentences. We name our approach **quick thoughts (QT)**, to mean efficient learning of thought vectors.

Our key contributions in this work are the following:

- We propose a simple and general framework for learning sentence representations efficiently. We train widely used encoder architectures an order of magnitude faster than previous methods, achieving better performance at the same time.

- We establish a new state-of-the-art for unsupervised sentence representation learning across several downstream tasks that involve understanding sentence semantics.

3.2 Related Work

We discuss prior approaches to learning sentence representations from labelled and unlabelled data.

Learning from Unlabelled corpora. Le and Mikolov (2014) proposed the paragraph vector (PV) model to embed variable-length text. Models are trained to predict a word given its context or words appearing in a small window based on a vector representation of the source document. Unlike most other methods, in this work sentences are considered as atomic units instead of as a compositional function of its words.

Encoder-decoder models have been successful at learning semantic representations. Kiros et al. (2015) proposed the skip-thought vectors model, which consists of an encoder RNN that produces a vector representation of the source sentence and a decoder RNN that sequentially predicts the words of adjacent sentences. Drawing inspiration from this model, Gan et al. (2016) explore the use of convolutional neural network (CNN) encoders. The base model uses a CNN encoder and reconstructs the input sentence as well as neighboring sentences using an RNN. They also consider a hierarchical version of the model which sequentially reconstructs sentences within a larger context.

Autoencoder models have been explored for representation learning in a wide variety of data domains. An advantage of autoencoders over context prediction models is that they do not require ordered sentences for learning. Socher et al. (2011) proposed recursive autoencoders which encode an input sentence using a recursive encoder and a decoder reconstructs the hidden states of the encoder. Hill et al. (2016) considered a de-noising autoencoder model (SDAE) where noise is introduced in a sentence by deleting words and swapping bigrams and the decoder is required to reconstruct the original sentence. Bowman et al. (2015) proposed a generative model of sentences based on a variational autoencoder.

Kenter et al. (2016) learn bag-of-words (BoW) representations of sentences by considering a conceptually similar task of identifying context sentences from candidates and evaluate their representations on sentence similarity tasks. Hill et al. (2016) introduced the FastSent model which uses a BoW representation of the input sentence and predicts the words appearing in context (and optionally, the source) sentences. The model is trained to predict whether a word appears in the target sentences. Arora et al. (2016) consider a weighted BoW model followed by simple post-processing and show that it performs better than BoW models trained on paraphrase data.

Jernite et al. (2017) use paragraph level coherence as a learning signal to learn representations. The following related task is considered in their work. Given the first three sentences of a paragraph, choose the next sentence from five sentences later in the paragraph.

Related to our objective is the local coherence model of Li and Hovy (2014) where a binary classifier is trained to identify coherent/incoherent sentence windows. In contrast, we only encourage observed contexts to be more plausible than contrastive ones and formulate it as a multi-class classification problem. We experimentally found that this relaxed constraint helps learn better representations.

Encoder-decoder based sequence models are known to work well, but they are slow to train on large amounts of data. On the other hand, bag-of-words models train efficiently by ignoring word order. We incorporate the best of both worlds by retaining flexibility of the encoder architecture, while still being able to to train efficiently.

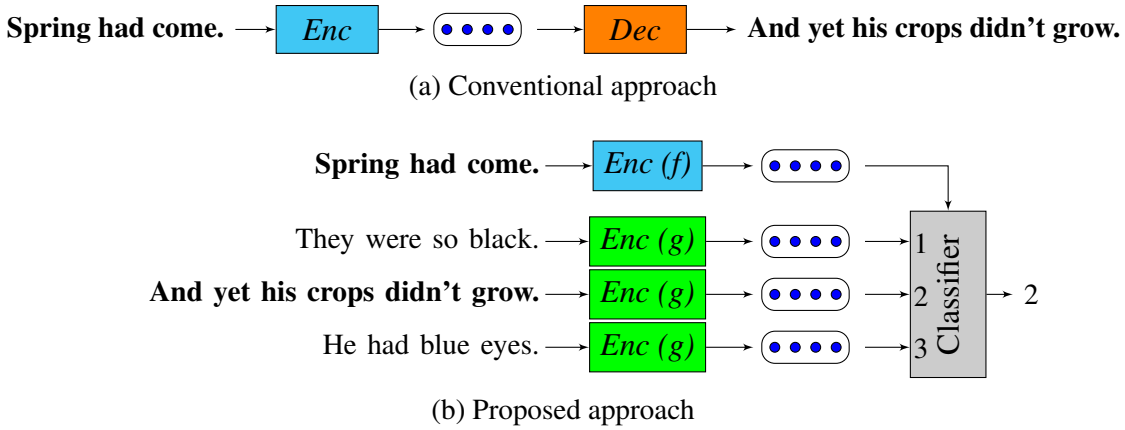
Structured Resources. There have been attempts to use labeled/structured data to learn sentence representations. Hill et al. (2016) learn to map words to their dictionary definitions using a max margin loss that encourages the encoded representation of a definition to be similar to the corresponding word. Wieting et al. (2015) and Wieting and Gimpel (2017) use paraphrase data to learn an encoder that maps synonymous phrases to similar embeddings using a margin loss. Hermann and Blunsom (2013) consider a similar objective of minimizing the inner product between paired sentences in different languages. Wieting et al. (2017) explore the use of machine translation to obtain more paraphrase data via back-translation and use it for learning paraphrastic embeddings.

Conneau et al. (2017) consider the supervised task of Natural language inference (NLI) as a means of learning generic sentence representations. The task involves identifying one of three relationships between two given sentences - entailment, neutral and contradiction. The training strategy consists of learning a classifier on top of the embeddings of the input pair of sentences. The authors show that sentence encoders trained for this task perform strongly on downstream transfer tasks.

3.3 Proposed Framework

The distributional hypothesis has been operationalized by prior work in different ways. A common approach is illustrated in Figure 3.1a, where an encoding function computes a vector representation of an input sentence, and then a decoding function attempts to generate the words of a target sentence conditioned on this representation. In the skip-thought model, the target sentences are those that appear in the neighborhood of the input sentence. There have been variations on the decoder such as autoencoder models which predict the input sentence instead of neighboring sentences (Hill et al., 2016) and predicting properties of a window of words in the input sentence (Le and Mikolov, 2014).

Instead of training a model to reconstruct the surface form of the input sentence or its neighbors, we take the following approach. Use the meaning of the current sentence to



(a) Conventional approach

(b) Proposed approach

Figure 3.1: Overview. (a) The approach adopted by most prior work where given an input sentence the model attempts to generate a context sentence. (b) Our approach replaces the decoder with a classifier which chooses the target sentence from a set of candidate sentences.

predict the meanings of adjacent sentences, where meaning is represented by an embedding of the sentence computed from an encoding function. Despite the simplicity of the modeling approach, we show that it facilitates learning rich representations.

Our approach is illustrated in figure 3.1. Given an input sentence, it is encoded as before using some function. But instead of generating the target sentence, the model chooses the correct target sentence from a set of candidate sentences. Viewing generation as choosing a sentence from all possible sentences, this can be seen as a discriminative approximation to the generation problem.

A key difference between these two approaches is that in figure 3.1, the model can choose to ignore aspects of the sentence that are irrelevant in constructing a semantic embedding space. Loss functions defined in a feature space instead of raw data space have been found to be attractive in recent work for similar reasons (Larsen et al., 2015; Pathak et al., 2017).

Formally described, let f and g be parametrized functions that take a sentence as input and encode it into a fixed length vector. Let s be a given sentence. Let S_{ctxt} be the set of sentences appearing in the context of s (for a particular context size) in the training data. Let S_{cand} be the set of candidate sentences considered for a given context sentence $s_{\text{ctxt}} \in S_{\text{ctxt}}$. In other words, S_{cand} contains a valid context sentence s_{ctxt} (ground truth) and many other non-context sentences, and is used for the classification objective as described below.

For a given sentence position in the context of s (e.g., the next sentence), the probability that a candidate sentence $s_{\text{cand}} \in S_{\text{cand}}$ is the correct sentence (i.e., appearing in the context of s) for that position is given by

$$p(s_{\text{cand}}|s, S_{\text{cand}}) = \frac{\exp[c(f(s), g(s_{\text{cand}}))]}{\sum_{s' \in S_{\text{cand}}} \exp[c(f(s), g(s'))]} \tag{3.1}$$

where c is a scoring function/classifier.

The training objective maximizes the probability of identifying the correct context sentences for each sentence in the training data D.

$$\sum_{s \in D} \sum_{s_{\text{ctxt}} \in S_{\text{ctxt}}} \log p(s_{\text{ctxt}} | s, S_{\text{cand}}) \tag{3.2}$$

The modeling approach encapsulates the Skip-gram approach of Mikolov et al. (2013) when words play the role of sentences. In this case the encoding functions are simple lookup tables considering words to be atomic units, and the training objective maximizes the similarity between the source word and a target word in its context given a set of negative samples.

Alternatively, we considered an objective function similar to the negative sampling approach of Mikolov et al. (2013). This takes the form of a binary classifier which takes a sentence window as input and classifies them as plausible and implausible context windows. We found objective (3.2) to work better, presumably due to the relaxed constraint it imposes. Instead of requiring context windows to be classified as positive/negative, it only requires ground-truth contexts to be more plausible than contrastive contexts. This objective also performed empirically better than a max-margin loss.

In our experiments, c is simply defined to be an inner product $c(u, v) = u^T v$. This was motivated by considering pathological solutions where the model learns poor sentence encoders and a rich classifier to compensate for it. This is undesirable since the classifier will be discarded and only the sentence encoders will be used to extract features for downstream tasks. Minimizing the number of parameters in the classifier encourages the encoders to learn disentangled and useful representations.

We consider f, g to have different parameters, although they were motivated from the perspective of modeling sentence meaning. Another motivation comes from word representation learning methods which use different sets of input and output parameters. Parameter sharing is further not a significant concern since these models are trained on large corpora. At test time, for a given sentence s we consider its representation to be the concatenation of the outputs of the two encoders $[f(s) \;\; g(s)]$.

Our framework allows flexible encoding functions to be used. We use RNNs as f and g as they have been widely used in recent sentence representation learning methods. The words of the sentence are sequentially fed as input to the RNN and the final hidden state is interpreted as a representation of the sentence. We use gated recurrent units (GRU) (Chung et al., 2015) as the RNN cell similar to Kiros et al. (2015).

3.4 Experimental Results

Evaluating sentence embeddings We evaluate our sentence representations by using them as feature representations for downstream NLP tasks. Alternative fine-grained evaluation tasks such as identifying word appearance and word order were proposed in Adi et al. (2017). Although this provides some useful insight about the representations, these tasks focus on the syntactic aspects of a sentence. We are more interested in assessing how well representations capture sentence semantics. Although limitations of these evaluations have been pointed out, we stick to the traditional approach of evaluating using downstream tasks.

Data Models were trained on the 7000 novels of the BookCorpus dataset (Kiros et al., 2015). The dataset consists of about 45M ordered sentences. We also consider a larger corpus for training: the UMBC corpus (Han et al., 2013), a dataset of 100M web pages crawled from the internet, preprocessed and tokenized into paragraphs. The dataset has 129M sentences, about three times larger than BookCorpus. For models trained from scratch, we used case-sensitive vocabularies of sizes 50k and 100k for the two datasets respectively.

Training A minibatch is constructed using a contiguous sets of sentences in the corpus. For each sentence, all the sentences in the minibatch are considered to be the candidate pool $S_{\textbf{cand}}$ of sentences for classification. This simple scheme for picking contrastive sentences performed as well as other schemes such as random sampling and picking nearest neighbors of the input sentence. Hyperparameters including batch size, learning rate, prediction context size were obtained using prediction accuracies (accuracy of predicting context sentences) on the validation set. A context size of 3 was used, i.e., predicting the previous and next sentences given the current sentence. We used a batch size of 400 and learning rate of 5e-4 with the Adam optimizer for all experiments. All our RNN-based models are single-layered and use GRU cells. Weights of the GRU are initialized using uniform Xavier initialization and gate biases are initialized to 1. Word embeddings are initialized from $U[-0.1, 0.1]$.

Tasks We evaluate the sentence representations on tasks that require understanding sentence semantics. The following classification benchmarks are commonly used: movie review sentiment (MR) (Pang and Lee, 2005), product reviews (CR) (Hu and Liu, 2004), subjectivity classification (SUBJ) (Pang and Lee, 2004), opinion polarity (MPQA) (Wiebe et al., 2005), question type classification (TREC) (Voorhees and Buckland, 2003) and paraphrase identification (MSRP) (Dolan et al., 2004). The semantic relatedness task on the SICK dataset (Marelli et al., 2014) involves predicting relatedness scores for a given pair of sentences that correlate well with human judgements.

The MR, CR, SUBJ, MPQA tasks are binary classification tasks. 10-fold cross validation

Model	Dim	Training time (h)	MR	CR	SUBJ	MPQA	TREC	MSRP (Acc)	MSRP (F1)	SICK r	SICK ρ	SICK MSE
GloVe BoW	300	-	78.1	80.4	91.9	87.8	85.2	72.5	81.1	0.764	0.687	0.425
Trained from scratch on BookCorpus data												
SDAE	2400	192	67.6	74.0	89.3	81.3	77.6	**76.4**	83.4	N/A	N/A	N/A
FastSent	<500	2*	71.8	78.4	88.7	81.5	76.8	72.2	80.3	N/A	N/A	N/A
ParagraphVec	<500	4*	61.5	68.6	76.4	78.1	55.8	73.6	81.9	N/A	N/A	N/A
uni-skip	2400	336	75.5	79.3	92.1	86.9	91.4	73.0	81.9	0.848	0.778	0.287
bi-skip	2400	336	73.9	77.9	92.5	83.3	89.4	71.2	81.2	0.841	0.770	0.300
combine-skip	4800	336†	76.5	80.1	**93.6**	87.1	**92.2**	73.0	82.0	0.858	0.792	0.269
combine-cnn	4800	-	77.2	80.9	93.1	**89.1**	91.8	75.5	82.6	0.853	0.789	0.279
uni-QT	2400	11	77.2	82.8	92.4	87.2	90.6	74.7	82.7	0.844	0.778	0.293
bi-QT	2400	9	77.0	83.5	92.3	87.5	89.4	74.8	82.9	0.855	0.787	0.274
combine-QT	4800	11†	**78.2**	**84.4**	93.3	88.0	90.8	76.2	**83.5**	**0.860**	**0.796**	**0.267**
Trained on BookCorpus, pre-trained word vectors are used												
combine-cnn	4800	-	77.8	82.1	93.6	**89.4**	92.6	76.5	83.8	0.862	0.798	0.267
MC-QT	4800	11	**80.4**	**85.2**	**93.9**	**89.4**	**92.8**	**76.9**	**84.0**	**0.868**	**0.801**	**0.256**
Trained on (BookCorpus + UMBC) data, from scratch and using pre-trained word vectors												
combine-QT	4800	28	81.3	84.5	94.6	89.5	92.4	75.9	83.3	0.871	0.807	0.247
MC-QT	4800	28	**82.4**	**86.0**	**94.8**	**90.2**	92.4	**76.9**	**84.0**	**0.874**	**0.811**	**0.243**

Table 3.1: Comparison of sentence representations on downstream tasks. The baseline methods are GloVe bag-of-words representation, De-noising auto-encoders and FastSent from Hill et al. (2016), the paragraph vector distributed memory model (Le and Mikolov, 2014), skip-thought vectors (Kiros et al., 2015) and the CNN model of Gan et al. (2016). Training times indicated using * refers to CPU trained models and † assumes concatenated representations are trained independently. Performance figures for SDAE, FastSent and ParagraphVec were obtained from Hill et al. (2016). Higher numbers are better in all columns except for the last (MSE). The table is divided into different sections. The bold-face numbers indicate the best performance values among models in the current and all previous sections. Best overall values in each column are underlined.

is used in reporting test performance for these tasks. The other tasks come with train/dev/test splits and the dev set is used for choosing the regularization parameter. We follow the evaluation scheme of Kiros et al. (2015) where feature representations of sentences are obtained from the trained encoders and a logistic/softmax classifier is trained on top of the embeddings for each task while keeping the sentence embeddings fixed. Kiros et al.'s scripts are used for evaluation.

3.4.1 Comparison against unsupervised methods

Table 3.1 compares our work against representations from prior methods that learn from unlabelled data. The dimensionality of sentence representations and training time are also

indicated. For our RNN based encoder we consider variations that are analogous to the skip-thought model. The uni-QT model uses uni-directional RNNs as the sentence encoders f and g. In the bi-QT model, the concatenation of the final hidden states of two RNNs represent f and g, each processing the sentence in a different (forward/backward) direction. The combine-QT model concatenates the representations (at test time) learned by the uni-QT and bi-QT models.

Models trained from scratch on BookCorpus While the FastSent model is efficient to train (training time of 2h), this efficiency stems from using a bag-of-words encoder. Bag of words provides a strong baseline because of its ability to preserves word identity information. However, the model performs poorly compared to most of the other methods. Bag-of-words is also conceptually less attractive as a representation scheme since it ignores word order, which is a key aspect of meaning.

The de-noising autoencoder (SDAE) performs strongly on the paraphrase detection task (MSRP). This is attributable to the reconstruction (autoencoding) loss which encourages word identity and order information to be encoded in the representation. However, it fails to perform well in other tasks that require higher level sentence understanding and is also inefficient to train.

Our uni/bi/combine-QT variations perform comparably (and in most cases, better) to the skip-thought model and the CNN-based variation of Gan et al. (2016) in all tasks despite requiring much less training time. Since these models were trained from scratch, this also shows that the model learns good word representations as well.

MultiChannel-QT Next, we consider using pre-trained word vectors to train the model. The MultiChannel-QT model (MC-QT) is defined as the concatenation of two bi-directional RNNs. One of these uses fixed pre-trained word embeddings coming from a large vocabulary ($\sim$ 3M) as input. While the other uses tunable word embeddings trained from scratch (from a smaller vocabulary $\sim$ 50k). This model was inspired by the multi-channel CNN model of Kim (2014) which considered two sets of embeddings. With different input representations, the two models discover less redundant features, as opposed to the uni and bi variations suggested in Kiros et al. (2015). We use GloVe vectors (Pennington et al., 2014) as pre-trained word embeddings. The MC-QT model outperforms all previous methods, including the variation of Gan et al. (2016) which uses pre-trained word embeddings.

UMBC data Because our framework is efficient to train, we also experimented on a larger dataset of documents. Results for models trained on BookCorpus and UMBC corpus pooled together ($\sim$ 174M sentences) are shown at the bottom of the table. We observe strict improvements on a majority of the tasks compared to our BookCorpus models. This shows

that we can exploit huge corpora to obtain better models while keeping the training time practically feasible.

Computational efficiency Our models are implemented in Tensorflow. Experiments were performed using cuda 8.0 and cuDNN 6.0 libraries on a GTX Titan X GPU. Our best BookCorpus model (MC-QT) trains in just under 11hrs (On both the Titan X and GTX 1080). Training time for the skip-thoughts model is mentioned as 2 weeks in Kiros et al. (2015) and a more recent Tensorflow implementation[1] reports a training time of 9 days on a GTX 1080. On the augmented dataset our models take about a day to train, and we observe monotonic improvements in all tasks except the TREC task. Our framework allows training with much larger vocabulary sizes than most previous models. Our approach is also memory efficient. The paragraph vector model has a big memory footprint since it has to store vectors of documents used for training. Softmax computations over the vocabulary in the skip-thought and other models with word-level reconstruction objectives incur heavy memory consumption. Our RNN based implementation (with the indicated hyperparamters and batch size) fits within 3GB of GPU memory, a majority of it consumed by the word embeddings.

3.4.2 Comparison against supervised methods

Model	MR	CR	SUBJ	MPQA	SST	TREC	MSRP		SICK
CaptionRep	61.9	69.3	77.4	70.8	-	72.2	-	-	-
DictRep	76.7	78.7	90.7	87.2	-	81.0	68.4	76.8	-
NMT En-to-Fr	64.7	70.1	84.9	81.5	-	82.8	-	-	-
InferSent	81.1	**86.3**	92.4	**90.2**	84.6	88.2	76.2	83.1	**0.884**
MC-QT	**82.4**	86.0	**94.8**	**90.2**	**87.6**	**92.4**	**76.9**	**84.0**	0.874

Table 3.2: Comparison against supervised representation learning methods on downstream tasks.

Table 3.2 compares our approach against methods that learn from labelled/structured data. The CaptionRep, DictRep and NMT models are from Hill et al. (2016) which are trained respectively on the tasks of matching images and captions, mapping words to their dictionary definitions and machine translation. The InferSent model of Conneau et al. (2017) is trained on the NLI task. In addition to the benchmarks considered before, we additionally also include the sentiment analysis binary classification task on Stanford Sentiment Treebank (SST) (Socher et al., 2013b).

[1] https://github.com/tensorflow/models/tree/master/research/skip_thoughts

Model	MR	CR	SUBJ	MPQA	SST	TREC	MSRP		SICK
Ensemble	82.7	**86.7**	**95.5**	90.3	**88.2**	93.4	78.5	85.1	**0.881**
Task specific methods									
AdaSent	**83.1**	86.3	**95.5**	**93.3**	-	92.4	-	-	-
CNN	81.5	85.0	93.4	89.6	88.1	**93.6**	-	-	-
TF-KLD	-	-	-	-	-	-	**80.4**	**85.9**	-
DT-LSTM	-	-	-	-	-	-	-	-	0.868

Table 3.3: Comparison against task-specific supervised models. The models are AdaSent (Zhao et al., 2015), CNN (Kim, 2014), TF-KLD (Ji and Eisenstein, 2013) and Dependency-Tree LSTM (Tai et al., 2015). Note that our performance values correspond to a linear classifier trained on fixed pre-trained embeddings, while the task-specific methods are tuned end-to-end.

The Infersent model has strong performance on the tasks. Our multichannel model trained on the (BookCorpus + UMBC) data outperforms InferSent in most of the tasks, with most significant margins in the SST and TREC tasks. Infersent is strong in the SICK task presumably due to the following reasons. The model gets to observes near paraphrases (entailment relationship) and sentences that are not-paraphrases (contradiction relationship) at training time. Furthermore, it considers difference features ($|u - v|$) and multiplicative features ($u * v$) of the input pair of sentences u, v during training. This is identical to the feature transformations used in the SICK evaluation as well.

Ensemble We consider ensembling to exploit the strengths of different types of encoders. Since our models are efficient to train, we are able to feasibly train many models. We consider a subset of the following model variations for the ensemble.

- Model type - Uni/Bi-directional RNN
- Word embeddings - Trained from scratch/Pre-trained
- Dataset - BookCorpus/UMBC

Models are combined using a weighted average of the predicted log-probabilities of individual models, the weights being normalized validation set performance scores. Results are presented in table 3.3. Performance of the best purely supervised task-specific methods are shown at the bottom for reference. Note that these numbers are not directly comparable with the unsupervised methods since the sentence embeddings are not fine-tuned. We observe that the ensemble model closely approaches the performance of the best supervised task-specific methods, outperforming them in 3 out of the 8 tasks.

COCO Retrieval								
	Image Annotation				Image Search			
Model	**R@1**	**R@5**	**R@10**	**Med r**	**R@1**	**R@5**	**R@10**	**Med r**
Pre-trained unsupervised sentence representations								
Combine-skip	33.8	67.7	82.1	3	25.9	60.0	74.6	4
Combine-cnn	34.4	-	-	3	26.6	-	-	4
MC-QT	**37.1**	**72.0**	**84.7**	**2**	**27.9**	**63.3**	**78.3**	**3**
Direct supervision of sentence representations								
DVSA	38.4	69.6	80.5	**1**	27.4	60.2	74.8	3
GMM+HGLMM	39.4	67.9	80.9	2	25.1	59.8	76.6	4
m-RNN	41.0	73.0	83.5	2	29.0	42.2	77.0	3
Order	**46.7**	**88.9**	-	2	**37.9**	**85.9**	-	**2**

Table 3.4: Image-caption retrieval. The purely supervised models are respectively from Karpathy and Fei-Fei (2015), Klein et al. (2015), Mao et al. (2014) and Vendrov et al. (2015). Best pre-trained representations and best task-specific methods are highlighted.

3.4.3 Image-Sentence Ranking

The image-to-caption and caption-to-image retrieval tasks have been commonly used to evaluate sentence representations in a multi-modal setting. The task requires retrieving an image matching a given text description and vice versa. The evaluation setting is identical to Kiros et al. (2015). Images and captions are represented as vectors. Given a matching image-caption pair (I, C) a scoring function f determines the compatibility of the corresponding vector representations v_I, v_C. The scoring function is trained using a margin loss which encourages matching pairs to have higher compatibility than mismatching pairs.

$$\sum_{(I,C)} \sum_{I'} \max\{0, \alpha - f(v_I, v_C) + f(v_I, v_{C'})\} + \sum_{(I,C)} \sum_{C'} \max\{0, \alpha - f(v_I, v_C) + f(v_{I'}, v_C)\} \quad (3.3)$$

As in prior work, we use VGG-Net features (4096-dimensional) as the image representation. Sentences are represented as vectors using the representation learning method to be evaluated. These representations are held fixed during training. The scoring function used in prior work is $f(x, y) = (Ux)^T(Vy)$ where U, V are projection matrices which project down the image and sentence vectors to the same dimensionality.

The MSCOCO dataset (Lin et al., 2014) has been traditionally used for this task. We use the train/val/test split proposed in Karpathy and Fei-Fei (2015). The training, validation and test sets respectively consist of 113,287, 5000, 5000 images, each annotated with 5 captions. Performance is reported as an average over 5 splits of 1000 image-caption pairs each from the test set. Results are presented in table 3.4. We outperform previous unsupervised pre-training methods by a significant margin, strictly improving the median retrieval rank for

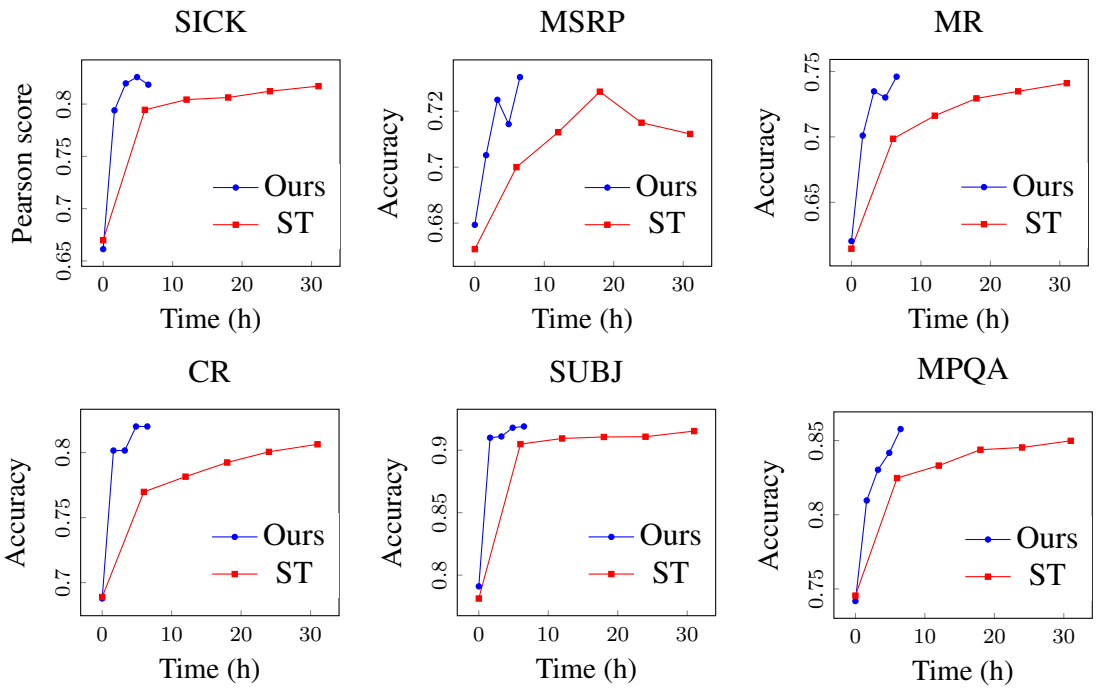


Figure 3.2: Same encoder architecture trained using our objective and Skip-thought (ST) objective and performance on downstream tasks is compared after a given number of hours.

both the annotation and search tasks. We also outperform some of the purely supervised task specific methods by some metrics.

3.4.4 Training Efficiency

To better assess the training efficiency of our models, we perform the following experiment. We train the same encoder architecture using our objective and the skip-thought (ST) objective and compare the performance after a certain number of hours of training. Since training the ST objective with large embedding sizes takes many days, we consider a lower dimensional sentence encoder for this experiment. We chose the encoder architecture to be a single-layer GRU Recurrent neural net with hidden state size $H = 1000$. The word embedding size was set to $W = 300$ and a vocabulary size of $V = 20,000$ words was used. Both models are initialized randomly from the same distribution. The models are trained on the same data for 1 epoch using the Adam optimizer with learning rate 5e-4 and batch size 400. For the low dimensional model considered, the model trained with our objective and ST objective take 6.5 hrs and 31 hrs, respectively.

The number of parameters for the two objectives are

- Ours: $6H(H + W + 1) + 2VW \approx 19.8M$ parameters
- ST: $9H(H + W + 1) + VW + 2HV \approx 57.7M$ parameters

Only the input side encoder parameters ($\approx 9.9M$ parameters) are used for the evaluation.

The 1000-dimensional sentence embeddings are used for evaluation. Figure 3.2 compares the performance of the two models on downstream tasks after x number of training hours. The speed benefits of our training objective is apparent from these comparisons.

The overall training speedup observed for our objective is 4.8x. Note that the output encoder was discarded for our model, unlike the experiments in the main text where the representations from the input and output encoders are concatenated. Further speedups can be achieved by training with encoders half the size and concatenating them (This is also parameter efficient).

Embedding size	Training time (h)	Performance
1600	4	84.57
2400	4.7	85.12
3200	6	85.74
4000	7.15	86.09
4800	8.5	86.43
5600	10.15	86.72

Table 3.5: Training time and performance for different embedding sizes. The reported performance is the mean accuracy over the classification benchmarks (MSRP, TREC, MR, CR, SUBJ, MPQA).

3.4.5 Representation size, training efficiency and performance

We explore the trade-off between training efficiency and the quality of representations by varying the representation size. We trained models with different representation sizes and evaluate them on the downstream tasks. The multi-channel model (MC-QT) was used for these experiments. Models were trained on the BookCorpus dataset.

Table 3.5 shows the training time and the performance corresponding to different embedding sizes. The training times listed here assume that the two component models in MC-QT are trained in parallel. The reported performance is an average over all the classification benchmarks (MSRP, TREC, MR, CR, SUBJ, MPQA). We note that the classifiers trained on top of the embeddings for downstream tasks differ in size for each embedding size. So it is difficult to make any strong conclusions about the quality of embeddings for the different sizes. However, we are able to reduce the embedding size and train the models more efficiently, at the expense of marginal loss in performance in most cases.

The 4800-dimensional Skip-thought model (Kiros et al., 2015) and Combine-CNN model (Gan et al., 2016) achieve mean accuracies of 83.75 and 85.33 respectively. We note

that our 1600-dimensional model and 3200-dimensional model are respectively better than these models, in terms of the mean performance across the benchmarks (We acknowledge that the Skip-thought model did not use pre-trained word embeddings). This suggests that high-quality models can be obtained even more efficiently by training lower-dimensional models on large amounts of data using our objective.

3.5 Conclusion

This chapter proposed a framework to learn generic sentence representations efficiently from large unlabelled text corpora. This simple approach learns richer representations than prior unsupervised and supervised methods, consuming an order of magnitude less training time. We establish a new state-of-the-art for unsupervised sentence representation learning methods on several downstream tasks. Exploring scalable approaches to learn data representations is key to exploit unlabelled data available in abundance.

The next sentence prediction task discussed in this chapter exploits a property of text called *coherence* to train representations. Coherence is the property that causes sentences to appear in a particular order. If the order is scrambled, the text is not readable anymore. In the next chapter I will present a more general version of the next sentence prediction problem that exploits this coherence signal naturally available in human authored text to train text representations.

CHAPTER IV

Representation Learning and Coherence Modeling via Text Ordering

Modeling the structure of coherent texts is a key NLP problem. The task of coherently organizing a given set of sentences has been commonly used to build and evaluate models that understand such structure. We propose an end-to-end unsupervised deep learning approach based on the set-to-sequence framework to address this problem. Our model strongly outperforms prior methods in the order discrimination task and a novel task of ordering abstracts from scientific articles. Furthermore, our work shows that useful text representations can be obtained by learning to order sentences. Visualizing the learned sentence representations shows that the model captures high-level logical structure in paragraphs. Our representations perform comparably to state-of-the-art pre-training methods on sentence similarity and paraphrase detection tasks.

4.1 Introduction

Modeling the structure of coherent texts is an important problem in NLP. A well-written text has a particular high-level logical and topical structure. The actual word and sentence choices and their transitions come together to convey the purpose of the text. Our primary goal is to build models that can learn such structure by arranging a given set of sentences to make coherent text.

Multi-document Summarization (MDS) and retrieval-based Question Answering (QA) involve extracting information from multiple documents and organizing it into a coherent summary. Since the relative ordering of sentences from different sources can be unclear, being able to automatically evaluate a particular order is essential. Barzilay and Elhadad (2002) discuss the importance of an ordering component in MDS and show that finding acceptable orderings can enhance user comprehension.

More importantly, by learning to order sentences we can model text coherence. It is difficult to explicitly characterize the properties of text that make it coherent. Ordering models attempt to understand these properties by learning high-level structure that causes sentences to appear in a specific order in human-authored texts. Automatic methods for evaluating human/machine generated text have great importance, with applications in essay scoring (Miltsakaki and Kukich, 2004; Burstein et al., 2010) and text generation (Park and Kim, 2015; Kiddon et al., 2016). Coherence models aid the better design of these systems.

Exploiting unlabelled corpora to learn semantic representations of data has become an active area of investigation. Self-supervised learning is a typical approach that uses information naturally available as part of the data as supervisory signals (Wang and Gupta, 2015; Doersch et al., 2015). Noroozi and Favaro (2016) attempt to learn visual representations by solving image jigsaw puzzles. Sentence ordering can be considered as a jigsaw puzzle in the language domain and an interesting question is whether we can learn useful textual representations by performing this task.

Our approach to coherence modeling is driven by recent success in capturing semantics using distributed representations and modeling sequences using Recurrent Neural Nets (RNN). RNNs are now the dominant approach to sequence learning and mapping problems. The Sequence-to-sequence (Seq2seq) framework (Sutskever et al., 2014) and its variants have fueled RNN based approaches to a range of problems such as language modeling, text generation, MT, QA, and many others.

In this work we propose an RNN-based approach to the sentence ordering problem which exploits the set-to-sequence framework of Vinyals et al. (2015a). A word-level RNN encoder produces sentence embeddings, and a sentence-level set encoder RNN iteratively attends to these embeddings and constructs a context representation. Initialized with this representation, a sentence-level pointer network selects the sentences sequentially.

The most widely studied task relevant to sentence ordering and coherence modeling is the order discrimination task. Given a document and a permuted version of it, the task involves identifying the more coherent ordering. Our proposed model achieves state-of-the-art performance on two benchmark datasets for this task, outperforming several classical approaches and recent data-driven approaches.

Addressing the more challenging task of ordering a given collection of sentences, we consider the novel and interesting task of ordering sentences from abstracts of scientific articles. Our model strongly outperforms previous work on this task. We visualize the learned sentence representations and show that our model captures high-level discourse structure. We provide visualizations that help understand what information in the sentences the model uses to identify the next sentence.

Finally, we demonstrate that our ordering model learns coherence properties and text representations that are useful for several downstream tasks including summarization, sentence similarity and paraphrase detection. In summary, our key contributions are as follows:

- We propose an end-to-end trainable model based on the set-to-sequence framework to address the problem of coherently ordering a collection of sentences.
- We consider the novel task of understanding structure in abstract paragraphs and demonstrate state-of-the-art results in order discrimination and sentence ordering tasks.
- We show that our model learns sentence representations that perform comparably to recent unsupervised pre-training methods on downstream tasks.

4.2 Related Work

Coherence modeling & sentence ordering. Coherence modeling and sentence ordering have been approached by closely related techniques. Many approaches propose a measure of coherence and formulate the ordering problem as finding an order with maximal coherence. Recurring themes from prior work include linguistic features, centering theory, local and global coherence.

Local coherence has been modeled by considering properties of local windows of sentences such as sentence similarity and transition structure. Lapata (2003) represent sentences by vectors of linguistic features and learn the transition probabilities between features of adjacent sentences. The Entity-Grid model (Barzilay and Lapata, 2008) captures local coherence by modeling patterns of entity distributions. Sentences are represented by the syntactic roles of entities appearing in the document. Features extracted from the entity grid are used to train a ranking SVM. These two methods are motivated from centering theory (Grosz et al., 1995), which states that nouns and entities in coherent discourses exhibit certain patterns.

Global models of coherence typically use HMMs to model document structure. The content model (Barzilay and Lee, 2004) represents topics in a particular domain as states in an HMM. State transitions capture possible presentation orderings within the domain. Words of a sentence are modeled using a topic-specific language model. The content model inspired several subsequent work to combine the strengths of local and global models. Elsner et al. (2007) combine the entity grid and the content model using a non-parametric HMM. Soricut and Marcu (2006) use several models as feature functions and define a log-linear model to assign probability to a text. Louis and Nenkova (2012) model the intentional structure in documents using syntax features.

Unlike previous approaches, we do not use any handcrafted features and adopt an embedding-based approach. Local coherence is taken into account by a next-sentence prediction component in our model, and global dependencies are naturally captured by an RNN. We demonstrate that our model can capture both logical and topical structure by several evaluation benchmarks.

Data-driven approaches Neural approaches have gained attention recently. Li and Hovy (2014) model sentences as embeddings derived from recurrent neural nets and train a feed-forward neural network that takes an input window of sentence embeddings and outputs a probability which represents the coherence of the sentence window. Coherence evaluation is performed by sliding the window over the text and aggregating the score. Li and Jurafsky (2016) study the same model in a larger scale task and also use a sequence-to-sequence approach in which the model is trained to generate the next sentence given the current sentence and vice versa. Nguyen and Joty (2017) learn to model coherence using a convolutional network that operates on the Entity-Grid representation of an input document. These models are limited by their local nature; our experiments show that considering larger contexts is beneficial.

Hierarchical RNNs for document modeling Word-level and sentence-level RNNs have been used in a hierarchical fashion for modeling documents in prior work. Li et al. (2015b) proposed a hierarchical autoencoder for generation and summarization. More relevant to our work is a similar model from Lin et al. (2015). A sentence-level RNN predicts the bag of words in the next sentence given the previous sentences and a word-level RNN predicts the word sequence conditioned on the sentence RNN hidden state. Our model has a hierarchical structure similar to these models, but takes a discriminative approach.

Combinatorial optimization with RNNs Vinyals et al. (2015a) equip sequence-to-sequence models with the ability to handle input and output sets, and discuss experiments on sorting, language modeling and parsing. This is called the *read*, *process*, *write* (or set-to-sequence) model. The *read* block maps input tokens to a fixed length vector representation. The *process* block is an RNN encoder which, at each time-step, attends to the input token embeddings and computes an attention readout, appending it to the current hidden state. The *write* block is an RNN which produces the target sequence conditioned on the representation produced by the process block. Their goal is to show that input and output orderings can matter in these tasks, which is demonstrated using small scale experiments. Our work exploits this framework to address the challenging problem of modeling logical and hierarchical structure in text. Vinyals et al. (2015b) proposed pointer-networks for combinatorial optimization problems where the output dictionary size depends on the number of input elements. We use

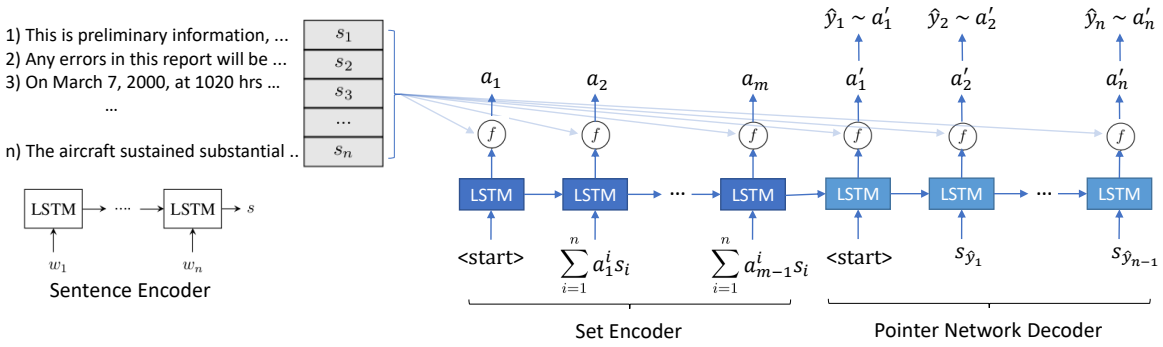


Figure 4.1: **Model Overview**: The input set of sentences are represented as vectors using a sentence encoder. At each time step, attention weights are computed for the sentence embeddings based on the current hidden state. The encoder uses the attention probabilities to compute the input for the next time-step and the decoder uses them for prediction.

a pointer-network as the decoder to sequentially pick the next sentence.

4.3 Approach

Our proposed model is inspired by the way a human would solve this task. First, the model reads the sentences to capture their meaning and the general context of the paragraph. Given this knowledge, the model tries to pick the sentences one by one sequentially till exhaustion.

Our model is based on the *read*, *process* and *write* framework of Vinyals et al. (2015a) briefly discussed in the previous section. We use the encoder-decoder terminology that is more common in the following discussion.

The model is comprised of a sentence encoder RNN, an encoder RNN and a decoder RNN (Fig. 4.1). The sentence encoder takes as input the words of a sentence s and computes an embedding representation of the sentence. Henceforth, we use s to refer to a sentence or its embedding interchangeably. The embeddings $\{s_1, s_2, ..., s_n\}$ of a given set of n sentences constitute the sentence memory, available to be accessed by subsequent components.

The encoder LSTM is identical to the originally proposed process block, defined by Eqs 4.1-4.4. At each time step the input to the LSTM is computed by taking a weighted sum over the memory elements, the weights being attention probabilities obtained using the previous hidden state as query (Eqs. 4.1, 4.2). This is iterated for a fixed number of times called the read cycles. Intuitively, the model identifies a soft input order to read the sentences. As described in Vinyals et al. (2015a) the encoder has the desirable property of being invariant to the order in which the sentence embeddings reside in the memory.

$$e^{t,i}_{\text{enc}} = f(s_i, h^t_{\text{enc}}); i \in \{1, ..., n\} \tag{4.1}$$

$$a^t_{\text{enc}} = \text{Softmax}(e^t_{\text{enc}}) \tag{4.2}$$

$$s^t_{\text{att}} = \sum_{i=1}^{n} a^{t,i}_{\text{enc}} s_i \tag{4.3}$$

$$h^{t+1}_{\text{enc}}, c^{t+1}_{\text{enc}} = \text{LSTM}(h^t_{\text{enc}}, c^t_{\text{enc}}, s^t_{\text{att}}) \tag{4.4}$$

The decoder is a pointer network that takes a similar form with a few differences (Eqs. 4.5-4.7). The LSTM takes the embedding of the previous sentence as input instead of the attention readout. At training time the correct order of sentences $(s_{o_1}, s_{o_2}, ..., s_{o_n}) = (x^1, x^2, ..., x^n)$ is known (o represents the correct order) and x^{t-1} is used as the input. At test time the predicted assignment $\hat{x}^{t-1}$ is used instead. The attention computation is identical to that of the encoder, but now $a^{t,i}_{\text{dec}}$ is interpreted as the probability for s_i being the correct sentence choice at position t, conditioned on the previous sentence assignments $p(S_t = s_i | S_1, ..., S_{t-1})$. The initial state of the decoder LSTM is initialized with the final hidden state of the encoder. x^0 is a vector of zeros.

$$h^t_{\text{dec}}, c^t_{\text{dec}} = \text{LSTM}(h^{t-1}_{\text{dec}}, c^{t-1}_{\text{dec}}, x^{t-1}) \tag{4.5}$$

$$e^{t,i}_{\text{dec}} = f(s_i, h^t_{\text{dec}}); i \in \{1, ..., n\} \tag{4.6}$$

$$a^t_{\text{dec}} = \text{Softmax}(e^t_{\text{dec}}) \tag{4.7}$$

Scoring Function We consider two choices for the scoring function f in Eqs. 4.1, 4.6. The first (Eq. 4.8) is a single hidden layer feed-forward net that takes s, h as inputs (W, b, W', b' are learnable parameters). The structure of f is similar to the window network of Li and Hovy (2014). While they used a local window of sentences to capture context, this scoring function exploits the entire history of sentences encoded in the RNN hidden state to score candidates for the next sentence.

$$f(s, h) = W' \text{tanh}(W[s\,;\,h] + b) + b' \tag{4.8}$$

We also consider a bilinear scoring function (Eq. 4.9). Compared to the previous scoring function, this takes a generative approach to regress the next sentence given the current hidden state $(Wh + b)$, enforcing that it be most similar to the correct next sentence. We observed that this scoring function led to better sentence representations (Sec. 4.4.4).

$$f(s, h) = s^T (Wh + b) \tag{4.9}$$

Contrastive Sentences In its vanilla form, we found that the set-to-sequence model tends to rely on certain word clues to perform the ordering task. To encourage holistic sentence

understanding, we add a random set of sentences to the sentence memory when the decoder makes classification decisions. This makes the problem more challenging for the decoder since now it has to distinguish between sentences that are relevant and irrelevant to the current context in identifying the correct sentence.

Coherence modeling We define the coherence score of an arbitrary partial/complete assignment $(s_{p_1}, ..., s_{p_k})$ to the first k sentence positions as

$$\sum_{i=1}^{k} \log p(S_i = s_{p_i} | S_{1,...,i-1} = s_{p_1,...,p_{i-1}}) \tag{4.10}$$

where $S_1, .., S_k$ are random variables representing the sentence assignment to positions 1 through k. The conditional probabilities are derived from the network. This is our measure of comparing the coherence of different renderings of a document. It is also used as a heuristic during decoding.

Training Objective The model is trained using the maximum likelihood objective

$$\max \sum_{x \in D} \sum_{t=1}^{|x|} \log p(x^t | x^1, ..., x^{t-1}) \tag{4.11}$$

where D denotes the training set and each training instance is given by an ordered document of sentences $x = (x^1, ..., x^{|x|})$.

4.4 Experimental Results

We first consider the order discrmination task that has been widely used in the literature for evaluating coherence models. We then consider the more challenging ordering problem where a coherent order of a given collection of sentences needs to be determined. We then demonstrate that our ordering model learns coherence properties useful for summarization. Finally, we show that our model learns sentence representations that are useful for downstream applications.

For all tasks discussed in this section we train the model with the maximum likelihood objective on the training data relevant to the task. We used the single hidden layer MLP scoring function for the order discrimination and sentence ordering tasks. Models are trained end-to-end. We use pre-trained 300 dimensional GloVe word embeddings (Pennington et al., 2014) to initialize word vectors. All LSTMs use a hidden layer size of 1000 and the MLP in Eq. 4.8 has a hidden layer size of 500. The number of read cycles in the encoder is set to 10. The same model architecture is used across all experiments. We used the Adam optimizer (Kingma and Ba, 2015) with batch size 10 and learning rate 5e-4 for learning. The model is regularized using early stopping. Hyperparameters were chosen using the validation set.

	Entity-Grid	HMM	Graph	Window (Recurrent)	Window (Recursive)	Seq2seq	Ours
Accidents	0.904	0.842	0.846	0.840	0.864	0.930	**0.944**
Earthquakes	0.872	0.957	0.635	0.951	0.976	0.992	**0.997**

Table 4.1: Mean Accuracy comparison on the Accidents and Earthquakes data for the order discrimination task. The reference models are Entity-Grid (Barzilay and Lapata, 2008), HMM (Louis and Nenkova, 2012), Graph (Guinaudeau and Strube, 2013), Window network (Li and Hovy, 2014) and sequence-to-sequence (Li and Jurafsky, 2016), respectively.

4.4.1 Order Discrimination

The ordering problem is traditionally formulated as a binary classification task: Given a reference paragraph and its permuted version, identify the more coherent one (Barzilay and Lapata, 2008).

The datasets widely used for this task in previous work are the Accidents and Earthquakes news reports. In each of these datasets the training and test sets include 100 articles and approximately 20 permutations of each article.

In Table 4.1 we compare our results with traditional approaches and recent data-driven approaches. The entity grid model provides a strong baseline on the Accidents dataset, only outperformed by our model and Li and Jurafsky (2016). On the Earthquakes data the window approach of Li and Jurafsky (2016) performs strongly. Our approach outperforms prior models on both datasets, achieving near perfect performance on the Earthquakes dataset.

While these datasets have been widely used, they are quite formulaic in nature and are no longer challenging. We hence turn to the more challenging task of ordering a given collection of sentences to make a coherent document.

4.4.2 Sentence Ordering

In this task we directly address the ordering problem. We do not assume the availability of a set of candidate orderings to choose from and instead find a good ordering from all possible permutations of the sentences.

The difficulty of the ordering problem depends on the nature of the text, as well as the length of paragraphs considered. Evaluation on text from arbitrary text sources makes it difficult to interpret the results, since it may not be clear whether to attribute the observed performance to a deficient model or ambiguity in next sentence choices due to many plausible orderings.

Text summaries are a suitable source of data for this task. They often exhibit a clear flow of ideas and have little redundancy. We specifically look at abstracts of conference papers

and research proposals. This data has several favorable properties. Abstracts usually have a particular high-level format - They begin with a brief introduction, a description of the problem and proposed approach and conclude with performance remarks. This would allow us to identify if the model can capture high-level logical structure. Second, abstracts have an average length of about 10, making the ordering task more accessible. This also gives us a significant amount of data to train and test our models.

We use the following sources of abstracts for this task.

- *NIPS Abstracts*. We consider abstracts from NIPS papers in the past 10 years. We parsed 3280 abstracts from paper pdfs and obtained 3259 abstracts after omitting erroneous extracts. The dataset was split into years 2005-2013 for training and 2014, 2015 respectively for validation, testing.

- *ACL Abstracts*. A second source of abstracts are papers from the ACL Anthology Network (AAN) corpus (Radev et al., 2009). We extracted 12,157 abstracts from the text parses using simple keyword matching for the strings 'Abstract' and 'Introduction'. We use all extracts of papers published up to year 2010 for training, year 2011 for validation and years 2012-2013 for testing.

- *NSF Abstracts*. We also used the NSF Research Award Abstracts dataset (Lichman, 2013). It comprises abstracts from a diverse set of scientific areas in contrast to the previous two sources of data and the abstracts are also lengthier, making this dataset more challenging. Years 1990-1999 were used for training, 2000 for validation and 2001-2003 for testing. We capped the parses of the abstracts to a maximum length of 40 sentences. Unsuccessful parses and parses of excessive length were discarded.

Further details about the data are provided in the supplement.

The following metrics are used to evaluate performance. *Accuracy* measures how often the absolute position of a sentence was correctly predicted. *Kendall's tau (τ)* is computed as $1 - 2 \cdot N / \binom{n}{2}$, where N is the number of pairs in the predicted sequence with incorrect relative order and n is the sequence length. Lapata (2006) discusses that this metric reliably correlates with human judgements.

The following baselines are used for comparison:

- **Entity Grid**. Our first baseline is the Entity Grid model of Barzilay and Lapata (2008). We use the Stanford parser (Klein and Manning, 2003) and Brown Coherence Toolkit[1] to derive Entity grid representations. A ranking SVM is trained to score

[1] bitbucket.org/melsner/browncoherence

correct orderings higher than incorrect orderings as in the original work. We used 20 permutations per document as training data. Since the entity grid only provides a means of feature extraction we evaluate the model in the ordering setting as follows. We choose 1000 random permutations for each document, one of them being the correct order, and pick the order with maximum coherence. We experimented with transitions of length at most 3 in the entity-grid.

- **Seq2seq**. The second baseline we consider is a sequence-to-sequence model which is trained to predict the next sentence given the current sentence. Li and Jurafsky (2016) consider similar methods and our model is the same as their uni-directional model. These methods were shown to yield sentence embeddings that have competitive performance in several semantic tasks in Kiros et al. (2015).

- **Window Network**. We consider the window approach of Li and Hovy (2014) and Li and Jurafsky (2016) which demonstrated strong performance in the order discrimination task as our third baseline. We adopt the same coherence score interpretation considered by the authors. In both the above models we consider a special embedding vector which is padded at the beginning of a paragraph and learned during training. This vector allows us to identify the initial few sentences during greedy decoding.

- **RNN Decoder**. Another baseline is our proposed model without the encoder. The decoder hidden state is initialized with zeros. We observed that using a special start symbol as for the other baselines helped obtain better performance with this model. However, a start symbol did not help when the model is equipped with an encoder as the hidden state initialization alone was good enough.

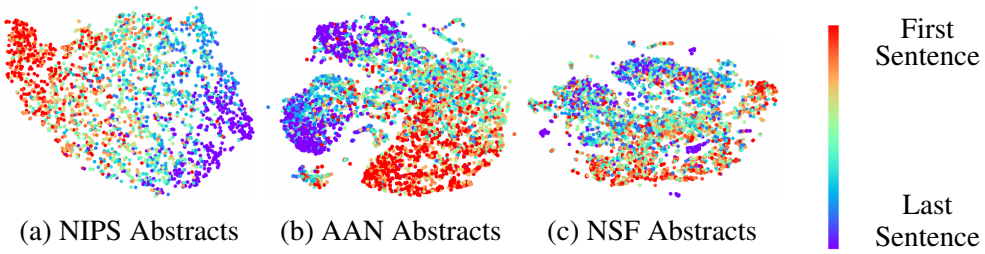
We do not place emphasis on the particular search algorithm in this work and thus use beam search with the coherence score heuristic for all models. A beam size of 100 was used. During decoding, sentence candidates that have been already chosen are pruned from the beam. All RNNs use a hidden layer size of 1000. For the window network we used a window size of 3 and a hidden layer size of 2000. We initialize all models with pre-trained GloVe word embeddings.

We assess the performance of our model against baseline methods in Table 4.2. The window network performs strongly compared to the other baselines. Our model does better by a significant margin by exploiting global context, demonstrating that global context is important in this task.

While the Entity-Grid model has been fairly successful for the order discrimination task in the past we observe that it fails to discriminate between a large number of candidates. One

	NIPS Abstracts		AAN Abstracts		NSF Abstracts	
	Accuracy	τ	Accuracy	τ	Accuracy	τ
Random	15.59	0	19.36	0	9.46	0
Entity Grid	20.10	0.09	21.82	0.10	-	-
Seq2seq (Uni)	27.18	0.27	36.62	0.40	13.68	0.10
Window network	41.76	0.59	50.87	0.65	18.67	0.28
RNN Decoder	48.22	0.67	52.06	0.66	25.79	0.48
Proposed model	**51.55**	**0.72**	**58.06**	**0.73**	**28.33**	**0.51**

Table 4.2: Comparison against prior methods on the abstracts data. Entity Grid, Seq2seq (Uni) and Window network are from Barzilay and Lapata (2008), Li and Jurafsky (2016), Li and Hovy (2014) respectively.



(a) NIPS Abstracts (b) AAN Abstracts (c) NSF Abstracts

Figure 4.2: t-SNE embeddings of representations learned by the model for sentences from the test set. Embeddings are color coded by the position of the sentence in the document it appears.

reason could be that the feature representation is less sensitive to local changes in sentence order (such as swapping adjacent sentences). The computational expense of obtaining parse trees and constructing grids on a large amount of data prohibited experimenting with this model on the NSF abstracts data.

The Seq2seq model performs worse than the window network. Interestingly, Li and Jurafsky (2016) observe that the Seq2seq model outperforms the window network in an order discrimination task on Wikipedia data. However, the Wikipedia data considered in their work is an order of magnitude larger than the datasets considered here, and that could have potentially helped the generative model. These models are also expensive during inference since they involve computing and sampling from word distributions.

Fig. 4.2 shows t-SNE embeddings of sentence representations learned by our sentence encoder. These are sentences from test sets, color coded by their positions in the source abstract. This shows that our model learns high-level structure in the documents, generalizing well to unseen text. The structure is less apparent in the NSF dataset due to its data diversity and longer documents. While approaches based on the Barzilay and Lee (2004) model

Model	ROUGE-1	ROUGE-2	ROUGE-L
	Summary length = 75b		
From scratch	18.29	47.56	12.79
Pre-train ordering model	18.77	50.32	13.25
	Summary length = 275b		
From scratch	35.82	10.67	33.69
Pre-train ordering model	36.47	10.99	34.27

Table 4.3: Comparison on extractive summarization between models trained from scratch and models pre-trained with the ordering task.

explicitly capture topics by discovering clusters in sentences, our neural approach implicitly discovers such structure.

4.4.3 Sentence Ordering and Summarization

In this section we show that sentence ordering models learn coherence properties useful for summarization. We consider a variation of our model where the model takes a set of sentences from several documents as input and sequentially picks summary sentences until it predicts a special 'stop' symbol. A key distinction between this model and recent work (Cheng and Lapata, 2016; Nallapati et al., 2016) is that the input order of sentences is assumed to be unknown, making it applicable to multi-document summarization.

We train a model from scratch to perform extractive summarization in the above fashion. We then consider a model that is pre-trained on the ordering task and is fine-tuned on the above task. The DailyMail and CNN datasets (Cheng and Lapata, 2016) were used for experimentation. We use DailyMail for pre-training purposes and CNN for fine-tuning and evaluation. The labels in DailyMail are not used. We compare ROUGE scores of the two models in Table 4.3 under standard evaluation settings.

We observe that the model pre-trained with the ordering task scores consistently better than the model trained from scratch. The results can be improved further by using larger news corpora. This shows that sentence ordering is an attractive unsupervised objective for exploiting large unlabelled corpora to improve summarization systems. It further shows that the coherence scores obtained from the ordering model correlates well with summary quality.

4.4.4 Learned Sentence Representations

One of the original motivations for this work is the question of whether we can learn high-quality sentence representations by learning to model text coherence. To address this

Model	SICK			MSRP	
	r	ρ	MSE	(Acc)	(F1)
Supervised	0.868	0.808	0.253	80.4	86.0
Uni-ST (Kiros et al., 2015)	0.848	0.778	0.287	73.0	81.9
Ordering model	0.807	0.742	0.356	72.3	81.1
+ BoW	0.842	0.775	0.299	74.0	81.9
+ Uni-ST	0.860	0.795	0.270	74.9	82.5

Table 4.4: Performance comparison for semantic similarity and paraphrase detection. The first row shows the best performing purely supervised methods. The last section shows our models.

question we trained our model on a large number of paragraphs using the BookCorpus dataset (Kiros et al., 2015).

To evaluate the quality of sentence embeddings derived from the model, we use the evaluation pipeline of Kiros et al. (2015) for tasks that involve understanding sentence semantics. These evaluations are performed by training a classifier on top of the embeddings derived from the model (holding the embeddings fixed) so that the performance is indicative of the quality of sentence representations. We present a comparison for the semantic relatedness and paraphrase detection tasks in Table 4.4. Results for only uni-directional versions of models are discussed here for a fair comparison. Similar to the skip-thought (ST) paper, we train two models - one predicting the correct order in the forward direction and another in the backward direction. The numbers shown for the ordering model were obtained by concatenating the representations from the two models.

Concatenating the above representation with the bag-of-words representation (using the fine-tuned word embeddings) of the sentence further improves performance. This is because the ordering model can choose to pay less attention to specific lexical information and focus on high-level document structure. Hence, the two representations capture complementary semantics. Adding ST features improves performance further. We observed that the bilinear scoring function and introducing contrastive sentences to the decoder improved the quality of learned representations significantly.

Our model has several key advantages over ST. ST has a word-level reconstruction objective and is trained with large softmax output layers. This limits the vocabulary size and slows down training (they use a vocabulary size of 20k and report two weeks of training). Our model achieves comparable performance and does not have such a word reconstruction component. We train with a vocabulary of 400k words; the above results are based on a training time of two days on a GTX Titan X GPU.

In this paper , we propose a new method for semantic class induction . First , we introduce a generative model of sentences , based on dependency trees and which takes into account homonymy . Our model can thus be seen as a generalization of Brown clustering . Second , we describe an efficient algorithm to perform inference and learning in this model . Third , we apply our proposed method on two large datasets (108 tokens , 105 words types) , and demonstrate that classes induced by our algorithm improve performance over Brown clustering on the task of semisupervised supersense tagging and named entity recognition .
Representation learning is a promising technique for discovering features that allow supervised classifiers to generalize from a source domain dataset to arbitrary new domains . We present a novel , formal statement of the representation learning task . We argue that because the task is computationally intractable in general , it is important for a representation learner to be able to incorporate expert knowledge during its search for helpful features . Leveraging the Posterior Regularization framework , we develop an architecture for incorporating biases into representation learning . We investigate three types of biases , and experiments on two domain adaptation tasks show that our biased learners identify significantly better sets of features than unbiased learners , resulting in a relative reduction in error of more than 16% for both tasks , with respect to state-of-the-art representation learning techniques.

Table 4.5: Visualizing salient words (Abstracts are from the AAN corpus).

4.4.5 Word Influence

We attempt to understand what text-level clues the model uses to perform the ordering task. Inspired by Li et al. (2015a), we use gradients of prediction decisions with respect to words of the correct sentence as a proxy for the salience of each word. We feed sentences to the decoder in the correct order and at each time step compute the derivative of the score e (Eq. 4.6) of the correct next sentence $s = (w_1, .., w_n)$ with respect to its word embeddings. The importance of word w_i in correctly predicting s as the next sentence is defined as $\|\frac{\partial e}{\partial w_i}\|$. We assume the hidden states of the decoder to be fixed and only back-propagate gradients through the sentence encoder.

Table 4.5 shows visualizations of two abstracts. Darker shades correspond to higher gradient norms. In the first example the model appears to be using the word clues 'first', 'second' and 'third'. A similar observation was made by Chen et al. (2016). In the second example we observe that the model pays attention to phrases such as 'We present', 'We argue', which are typical of abstract texts. It also focuses on the word 'representation'

appearing in the first two sentences. These observations link to centering theory which states that entity distributions in coherent discourses exhibit certain patterns. The model implicitly learns these patterns with no syntax annotations or handcrafted features.

4.5 Conclusion

This work investigated the challenging problem of coherently organizing a set of sentences. Our RNN-based model performs strongly compared to baselines and prior work on sentence ordering and order discrimination tasks. We further demonstrated that it captures high-level document structure and learns useful sentence representations when trained on large amounts of data. Our approach to the ordering problem deviates from most prior work that use handcrafted features. However, exploiting linguistic features for next sentence classification can potentially further improve performance. Entity distribution patterns can provide useful features about named entities that are treated as out-of-vocabulary words. The ordering problem can be further studied at higher-level discourse units such as paragraphs, sections and chapters.

The previous and current chapters discussed methods to train text representations from large scale unlabelled text. Given these pre-trained representation models, the next three chapters explore how to train models for new tasks and domains with limited supervision. In the next chapter, I present a meta-learning algorithm to adapt the transformer architecture, which has been widely successful for training representations, to the few-shot setting.

CHAPTER V

Few-shot Sequence Learning with Transformers

Few-shot algorithms aim at learning new tasks provided only a handful of training examples. In this work we investigate few-shot learning in the setting where the data points are sequences of tokens and propose an efficient learning algorithm based on Transformers. In the simplest setting, we append a token to an input sequence which represents the particular task to be undertaken, and show that the embedding of this token can be optimized on the fly given few labeled examples. Our approach does not require complicated changes to the model architecture such as adapter layers nor computing second order derivatives as is currently popular in the meta-learning and few-shot learning literature. We demonstrate our approach on a variety of tasks, and analyze the generalization properties of several model variants and baseline approaches. In particular, we show that compositional task descriptors can improve performance. Experiments show that our approach works at least as well as other methods, while being more computationally efficient.

5.1 Introduction

The problem of learning a classifier from a handful examples has received considerable attention in the vision domain under the name of *few-shot learning* (Fink, 2005; Fei-Fei et al., 2006). However, less work exists in the space of few-shot problems involving *discrete* sequences, such as sequences of discrete actions in reinforcement learning or sequences of words in natural language processing. In this work, we study the problem of sequence classification and modeling in the few-shot regime. Specifically, we assume there are several training tasks available for learning and, at test time, we are interested in performing few-shot adaptation to a given new task.

Transformers (Vaswani et al., 2017a) have been very successful at modeling discrete sequences (Barrault et al., 2019; Devlin et al., 2019; Parisotto et al., 2019). Further, they have been shown to use context tokens appended to an input to adapt their generations or to

switch between different tasks (Lample et al., 2019; Shen et al., 2019; Zellers et al., 2019; Keskar et al., 2019). Thus, one might hope that such context tokens could be effectively used in the meta-learning setting for discrete sequences.

In this work, we show that this is indeed the case. Our approach to few-shot learning introduces a set of task specific parameters (a *task embedding*), in addition to the parameters of the model that are shared among all tasks. Unlike other approaches that require architectural changes (Houlsby et al., 2019), task embeddings are simply fed to the input of the transformer. Learning a new task consists of inferring an appropriate task embedding for the task, leaving the shared model parameters intact. Towards this end, we propose a simple training algorithm where the task embedding is found via gradient based optimization, which is simpler and computationally less expensive than second order optimization methods (Finn et al., 2017; Zintgraf et al., 2019).

To summarize, our contributions in this work are as follows. First, we show that a simple alternating-minimization approach for few-shot learning works well in combination with the transformer architecture. Second, we show that a simple yet effective way to condition the transformer with task information is via input conditioning (i.e., feeding task information as input to the transformer); this naturally extends to compositional task information. Third, we introduce a battery of synthetic sequence classification and modeling tasks to benchmark in a controlled setting various baseline approaches and model variants for few shot learning of discrete sequences. And finally, we demonstrate that the proposed approach offers a better trade-off between few-shot performance and run time cost compared to other baselines, including meta-learning approaches.

5.2 Problem Definition

We assume a distribution $p_{\text{data}}(\mathcal{T})$ over tasks from which disjoint sets of training, validation and test sets of tasks are drawn. The set of training tasks is denoted by $\{\mathcal{T}_i^{\text{train}}\}_{i=1}^{N}$, where each task $\mathcal{T}_i^{\text{train}}$ has an associated set of training examples $\{(x_j^i, y_j^i)_{j=1}^{N_i}\}$. Validation and test tasks are defined similarly, except each test task only has k training examples.

We focus on two types of tasks that involve discrete sequences as inputs and outputs: sequence classification and transduction. In *sequence classification*, the inputs x are sequences and the output y is a discrete categorical label. In *sequence transduction*, each task consists of modeling the joint distribution of a sequence y, conditioned on some input context sequence x. The performance metrics for these settings are respectively accuracy and perplexity, averaged across the test tasks.

Our training and inference protocol is as follows. We use the training tasks to learn the model parameters and the validation tasks to determine hyperparameters. The optimal pa-

rameters and hyperparameters identified are used for evaluating average model performance on test tasks. This involves first (optionally) training on the small set of training examples accompanied by a test task, followed by testing on the corresponding test set.

5.3 Approach

5.3.1 Architecture

In this work, we explore an adaptation of transformers to the few-shot regime. Previous works have shown that the behavior of a transformer model can be conditioned by appending special tokens describing the task to be performed on the input sequence (Lample et al., 2019; Zellers et al., 2019). We append a *task embedding* vector which represents information about the task of interest to the input sequence of token embeddings. We intend to control the overall behavior of the model for a particular task by altering the task embeddings while keeping the rest of the model parameters intact.

For classification tasks, we use a transformer encoder similar to the BERT model (Devlin et al., 2019). A classification head sits on the final layer representation of a special token at the beginning of the sequence. We replace this special token with the task embedding vector z in our model. We use a transformer decoder architecture for the transduction tasks and append the task embedding vector to the input sequence similar to the classification setting.

In both settings we compute a log-likelihood of the form $\log p(y|x, z; \theta)$, where x is the input sequence, z is the task embedding and θ the model parameters. In the classification setting y is a single categorical value. In the sequence transduction setting, y is a sequence and the log likelihood decomposes as the sum of the conditional log-likelihood terms via the chain rule of probability theory: $\log p(y|x, z; \theta) = \sum_i \log p(y_i|y_{i-1}, \cdots, y_1, x, z; \theta)$.

Note that in practical applications θ can be high-dimensional, in the order of hundreds of millions. Our goal is to alleviate overfitting in the few-shot regime by adapting only z to learn a new task, where z is a small vector with at most a few hundred components. Next, we describe how we learn the model parameters θ and how we estimate the task embedding z for a given task.

5.3.2 Training and Inference Algorithm

We train our models with an alternating-minimization scheme similar to Maurer et al. (2013) and Kumar and Daume III (2012), that can be considered a simplification of the CAVIA approach in Zintgraf et al. (2019) (or as a refinement of the "first-order" method in that work). See Algorithm 1 for pseudo-code. We separate the weights of the network defining the model into shared weights θ, and per-task weights, as in CAVIA. In our case, the per-task

Algorithm 1: TAM for k-shot Learning

Input : Training tasks $\mathcal{T}_1^{\text{train}}, ..., \mathcal{T}_N^{\text{train}}$
Output : Model parameters θ

1 **repeat**
2 Sample a training task: $\mathcal{T}_i^{\text{train}}$;
3 Sample $N_i \geq k$ training examples from the task $\{(x^j, y^j)_{j=1,\cdots,N_i}\} \sim \mathcal{T}_i^{\text{train}}$;
4 Initialize: $z_{\mathcal{T}_i^{\text{train}}} = 0, \Delta\theta = 0$;
5 **while** *loss improves and max number of updates not reached* **do**
6 $z_{\mathcal{T}_i^{\text{train}}} \leftarrow z_{\mathcal{T}_i^{\text{train}}} - \nabla_{z_{\mathcal{T}_i^{\text{train}}}} \sum_j -\log p(y^j|x^j, z_{\mathcal{T}_i^{\text{train}}}; \theta)$;
7 $\Delta\theta \leftarrow \Delta\theta - \nabla_\theta \sum_{j=1}^{N_i} -\log p(y^j|x^j, z_{\mathcal{T}_i^{\text{train}}}; \theta)$
8 $\theta \leftarrow \theta + \Delta\theta$
9 **until** *max training iterations*;

weights form the embedding z, one for each task; while all other parameters θ are shared.

Given a few examples from the training task $\mathcal{T}_i^{\text{train}}$ (line 3), we alternate training $z_{\mathcal{T}_i^{\text{train}}}$ (the task embedding of the $\mathcal{T}_i^{\text{train}}$ task) for a few gradient descent steps keeping θ fixed (see line 5 and 6), and then update θ based on the optimal task embedding. In practice, however, we found it helpful to update θ based on gradients accumulated for the intermediate values of the task embedding encountered in the inner loop optimization (line 7). We surmise that this optimization choice helps the model finding better task embeddings as the whole parameter vector θ is updated to account for this search. Task embedding gradient updates (line 6) are performed until the loss no longer improves or the maximum number of update steps has been reached. Note that unlike prior methods such as MAML or CAVIA we do not backpropagate gradients through an optimization process, which simplifies and speeds up our optimization. We call our method, transformer trained with Alternating Minimization (TAM) – although the alternating minimization algorithm could be applied to other architectures as well. At test time, given a new task $\mathcal{T}^{\text{test}}$, $z_{\mathcal{T}^{\text{test}}}$ is trained with a few steps of gradient descent (similar to line 6), with all other parameters held fixed. Since TAM is trained to optimize task embeddings on the fly, we expect it to find good embeddings of the new task at test time as well.

5.4 Related work

Few-shot learning and meta-learning There is now a vast literature on learning methods designed for quickly adapting to new settings. At a coarse level, one can consider classes of methods that adapt the learning *algorithm* based on the task (and so are "meta-learners") (Schmidhuber, 1987; Hochreiter et al., 2001; Andrychowicz et al., 2016; Finn

et al., 2017; Nichol and Schulman, 2018), or describe *model architectures* that can adapt to learn sample-efficiently over a task distribution (Vinyals et al., 2016; Snell et al., 2017). Many methods have elements of both of these, e.g. Mishra et al. (2018); Rusu et al. (2019); Zintgraf et al. (2019).

The method we describe in this work can be considered squarely in the class of model architectures for sample efficient learning. It is a descendant of Hinton and Plaut (1987); Schmidhuber (1992); Ba et al. (2016) and is closely related to Rusu et al. (2019); Zintgraf et al. (2019) in that we pick a subset of the weights of the model that are task specific (the "fast" weights), and update them using the training examples for a specific task; but update the other ("slow") weights on all training examples for all tasks. Our approach is closest to Zintgraf et al. (2019), but differs in the way the fast weights are used by the model, and because we do not use higher-order gradients for the slow weights, instead we use an alternating minimization type update.

Task transfer for transformers Our approach is also related to other recent work in natural language processing. We leverage the particular structure of the transformer architecture (Vaswani et al., 2017a), which has been successful in many NLP tasks. Several works have shown that adding a token to an input can be used to switch between different tasks (Lample et al., 2019; Shen et al., 2019; Zellers et al., 2019; Keskar et al., 2019). Transformer language models trained on large corpora have also been recently shown to have impressive few-shot learning capabilities (Brown et al., 2020).

More generally, with the success of methods based on pretraining transformer models (Devlin et al., 2019), and finetuning on target tasks, there have been several works discussing how to adapt a pre-trained model without full finetuning (Houlsby et al., 2019; Stickland and Murray, 2019) but their focus has been on reducing the number of parameters subject to optimization at finetuning time as opposed to reducing the number of examples as in this study.

5.5 Experiments

5.5.1 Model and Training Details

In the classification setting, TAM is a bidirectional transformer that takes the input sequence x and the task embedding z as input, and outputs a distribution over classes. In the sequence transduction setting, TAM is a transformer decoder with a causal attention mechanism and takes as additional input the output sequence y up to the token before the last. In this case the model is trained to predicted the sequence y at the last $|y|$ (length of sequence y) time steps. Since model parameters are shared across tasks, TAM needs to leverage the task

embedding to perform the tasks well. Both classification and transduction models are trained with cross-entropy loss.

Unless otherwise specified our transformer has 4 layers with an embedding size of 128. We use the Adam optimizer (Kingma and Ba, 2015) for both outer and inner loop optimization. The maximum number of task embedding optimization steps is set to 25 during training. We train a single model using N samples at training time, treating N as a hyperparameter and apply it to k-shot problems with different values of k at test time. The size of the task embedding was set to match the embedding dimension of the transformer (128). We discuss more details about hyperparameter choices and how they influence model performance in section 5.5.6.

5.5.2 Baselines

Task-Agnostic transformer This baseline uses the same architecture as TAM but is not informed about the existence of different tasks at training time, i.e., no task embedding is fed at the input. At test time, the model is fine-tuned on the k training examples from the test task.

Multitask transformer This is a transformer that is conditioned on the current task both in the classification and transduction settings. It is identical to TAM except all parameters including task embeddings are trained by standard back-propagation, without any alternating minimization.

Matching Networks (Vinyals et al., 2016) We consider Matching Networks only in our classification setting, as it is not straightforward to use it for transduction. We use a transformer to model the similarity between a query instance and support set instance which takes the concatenation of the two sequences as input and outputs a similarity score.

SNAIL (Mishra et al., 2018) This model is similar to the task-agnostic transformer except the input is augmented with the concatenation of all input-output training pairs. For both Matching Networks and SNAIL, we construct training episodes by sampling k training examples to define a task, to match the test scenario. We train different models for each k-shot problem. Both Matching networks and SNAIL are trained using the multi-task training loss and applied to test tasks without any finetuning.

MAML (Finn et al., 2017) All model parameters are trained using MAML, with the same model architecture as TAM. The entire model is fine-tuned on test tasks.

CAVIA (Zintgraf et al., 2019) Similar to TAM, CAVIA has a set of task-specific parameters and shared parameters. The training algorithm is similar to MAML, but inner loop updates are performed on the task-specific parameters as opposed to the entire model.

5.5.3 Sequence Classification and Transduction

Most prior work on few-shot learning have focused on computer vision benchmarks such as Omniglot Lake (2019) and Mini-ImageNet Vinyals et al. (2016); Ravi and Larochelle (2016). In the sequential data setting, Bao et al. (2019) recently constructed synthetic benchmarks from existing text datasets but the number of tasks is rather limited. In this work we construct a new set of benchmarks involving synthetic sequential data, allowing us to evaluate models in a more controlled setting after training on a larger number of tasks.

Synthetic Benchmarks We construct a synthetic few-shot *classification* benchmark as follows. The benchmark consists of tasks that involve a non-negative integer sequence as input and a discrete label as output. A task is constructed by applying a sequence of mathematical transformations to input sequences as follows: Element-wise transform (T_1) $\rightarrow$ Subsequence extraction (T_2) $\rightarrow$ Labeling function (T_3). The arrows indicate function composition and the sequence of transformations maps an input sequence to a single integer. The transformations are defined as $T_1 \in S_1, T_2 \in S_2, T_3 \in S_3$ where, $S_1 = \{$*mul* v, *add* v, *div* v, *mod* $v\}$; $S_2 = \{$*(not) multiple of* v, *(not) greater than* v, *(do not) have exactly* v *divisors*$\}$; $S_3 = \{$*count, min, max, mean, median, mode, first, last, max-min, middle*$\}$, where $v \in \{1 \cdots n\}$ for some integer n. We randomly generate a large number of sequences X of integers from $\{0 \cdots N\}$. We apply the transformation sequence T_1, T_2, T_3 to these sequences $x \in X$ and get the corresponding outputs $T_3(T_2(T_1(x)))$. The C most frequent outputs are then defined to be the C classes of interest. We obtain a uniform amount of data from each class and discard input sequences for which the output does not belong to one of the chosen C classes. Cases where at least C distinct outputs cannot be obtained are discarded. These C classes then constitute a C-way classification task. An example task is `mul 2` $\rightarrow$ `less than 5` $\rightarrow$ `count`, where the goal is to count the number of input elements which, when multiplied by 2, are less than 5 (i.e., count number of input integers less than 3). The semantics of each of the transforms are defined in the appendix. We set $C = 4$ in our experiments. Vocabulary size and input sequence length are set to 12 and 5, respectively. Combinations of transforms that have identical input-output relationship are identified and removed during task construction. All tasks are thus unique in terms of input-output mapping.

We also construct two sequence transduction benchmarks. The first benchmark is constructed in a way similar to the classification tasks where we consider a sequence of transformations mapping an input sequence to an output sequence $T_1 \rightarrow T_2 \rightarrow T_3$, where $T_1 \in S_1, T_2 \in S_2, T_3 \in S_3$; $S_1 = \{$*mul* v, *add* v, *div* v, *mod* $v\}$; $S_2 = \{$*replace* v *with* v', *replace* x_i *with* $f(x_i, x_j)$ $\}$; $S_3 = \{$*sort ascending, sort descending, reverse, swap*(x_i, x_j), *shift*

Model	Sequence Classification				Sequence Transduction				Path Finding			
	1	5	10	20	1	5	10	20	1	5	10	20
Task Agnostic	40.50 ±1.73	64.75 ±1.29	74.25 ±1.29	82.50 ±1.58	6.27 ±0.17	5.26 ±0.04	4.71 ±0.05	4.01 ±0.07	3.17 ±0.27	1.75 ±0.03	1.55 ±0.02	1.39 ±0.01
Multitask	38.75 ±0.96	66.00 ±0.82	77.50 ±1.29	87.50 ±0.58	13.80 ±3.36	6.80 ±0.26	5.18 ±1.01	2.91 ±0.49	6.39 ±1.96	1.98 ±0.12	1.64 ±0.05	1.44 ±0.02
Matching Network	**43.00** ±1.15	58.75 ±2.45	64.50 ±1.83	67.00 ±1.41	–				–			
SNAIL	**43.00** ±1.41	44.00 ±2.00	68.25 ±1.26	67.50 ±4.43	**2.48** ±0.38	3.80 ±0.38	4.98 ±0.03	4.11 ±2.94	2.63 ±0.27	1.95 ±0.25	3.47 ±1.56	3.04 ±1.01
MAML	39.60 ±0.55	63.40 ±0.89	71.80 ±0.84	78.80 ±0.84	6.73 ±0.16	5.84 ±0.2	5.19 ±0.08	4.20 ±0.10	5.49 ±0.85	2.05 ±0.03	1.65 ±0.01	1.44 ±0.01
CAVIA	**43.00** ±0.58	**78.00** ±1.26	87.00 ±0.50	91.00 ±0.58	11.05 ±2.94	**2.75** ±0.51	1.78 ±0.14	1.53 ±0.06	2.21 ±0.21	1.31 ±0.03	1.25 ±0.03	1.21 ±0.02
TAM	40.50 ±0.82	75.50 ±0.50	**89.50** ±0.58	**94.50** ±0.82	8.47 ±1.42	2.92 ±0.67	**1.47** ±0.18	**1.15** ±0.03	**1.82** ±0.08	**1.27** ±0.01	**1.22** ±0.01	**1.17** ±0.01

Table 5.1: k-shot sequence classification and sequence transduction experiments on our three benchmarks for $k \in \{1, 5, 10, 20\}$. The metric for sequence classification is average accuracy on test tasks (higher is better). On the transduction tasks, the performance metric is average perplexity on test tasks (lower is better). Random performance is at 25% accuracy (classification) and 12 perplexity points (other two tasks). Entries in smaller font are error bars, and they are estimated on 4 trials varying the model initialization.

right $v\}$, and v, v', i, j are integers chosen at random, x_p represents the element at position p in the input sequence, f is a mathematical function (Eg: $f(a, b) \in \{a+b, \text{abs}(a-b), b, \cdots\}$). An example task is `add 2` $\rightarrow$ `replace 2 with 1` $\rightarrow$ `reverse`, and an (input, output) sample drawn from this task is: $([0, 5, 0, 3, 6], [8, 5, 1, 7, 1])$.

Our second transduction benchmark is a path finding task in a grid world. A task is defined by start and end positions in a square grid of size $N \times N$. Given the locations of obstacles in this grid, the objective of the task is to find the shortest path connecting start and end positions that avoids the obstacles. The source and target sequences correspond to the locations of obstacles and optimal path from start to end position avoiding the obstacles, respectively.

We use 500, 16, 64 tasks respectively for training, validation and testing for all three setups. Tasks are unique and randomly assigned to these sets, in other words we test generalization under the condition of distributional match between the training and the test set. Each training task has 500 examples.

Results Table 5.1 reports the results on this benchmark. In the extreme few-shot setting ($k = 1$), all methods perform poorly, although memory based methods such as matching networks and SNAIL fare the best. However, they start performing relatively worse when

more labelled data is available, where fine-tuning part or all of the model parameters could be beneficial. Both SNAIL and matching networks sometimes perform *absolutely* worse when more labeled examples are present, suggesting they are failing to effectively use their memory when confronted with longer sequences. Fine-tuning the whole model, particularly in the multitask setting, works remarkably well for larger values of k, although the best performance is achieved by TAM, suggesting the need for sample efficient task adaptation methods. For $k > 1$, TAM performs comparably or better than all baselines, including MAML and CAVIA. Furthermore, TAM is more efficient to train than CAVIA (see section 5.5.6).

5.5.4 Compositional Task Representations

Compositional reasoning is arguably an important skill for few-shot learning (Lake, 2019; Purushwalkam et al., 2019). The underlying assumption is that there exist primitive skills which can be learned and combined together to solve new tasks. If a learner can leverage the compositional structure of the learning task, it may learn with even less labeled data.

In this section we assess how much better TAM works when we expose the compositional structure of the tasks described in §5.5.3. Specifically, we assess the ability to learn new tasks which are composed of primitives, some of which were unseen during training. To present an example from the classification setting, assume the models know that tasks are composed of three transforms $T_1 \in S_1, T_2 \in S_2, T_3 \in S_3$. We henceforth refer to the elements of $S_1 \cup S_2 \cup S_3$ as *primitives*. Further assume the model never saw the *add* k primitive during training. Given a new test task for which $T_1 =$ *add* 3 (and T_2, T_3 are known primitives seen during training), we expect the model to infer the concept of *add* from the few training examples of the test task.

Task Construction In the compositional setting, we provide models with information about the primitives used to construct the task. For the classification and transduction tasks, the training and test tasks are constructed as follows. Assume the set of primitives available for the three transforms to be S_1, S_2, S_3. We hold out a subset of primitives S_1', S_2', S_3' respectively from each of these three sets, which shall constitute the *unseen* primitives. The training tasks are made up of primitives from $S_1 - S_1', S_2 - S_2', S_3 - S_3'$, which we will refer to as *seen* primitives. The test tasks are made up of seen and unseen primitives where exactly one primitive is unseen (For instance, $T_1 \in S_1 - S_1', T_2 \in S_2', T_3 \in S_3 - S_3'$). Model performance is averaged over multiple (8) different choices of S_1', S_2', S_3'.

We also define a compositional path-finding task as follows. In addition to finding the optimal path from start, end positions while avoiding obstacles, we now require the path to lie on a specified way-point. The locations of the start, end and way points thus define the

Model	Sequence Classification				Sequence Transduction				Path Finding			
	1	5	10	20	1	5	10	20	1	5	10	20
Multitask	57.50 ±3.51	74.00 ±5.48	81.00 ±4.24	88.5 ±2.08	43.32 ±10.87	7.50 ±0.43	3.48 ±0.12	2.16 ±0.05	3.08 ±0.61	1.62 ±0.33	1.32 ±0.05	1.23 ±0.02
Matching Network	61.25 ±4.35	69.50 ±5.45	72.25 ±6.08	67.5 ±5.92	–				–			
SNAIL	**63.5** ±4.80	71.75 ±4.86	76.25 ±2.75	80.25 ±2.87	7.00 ±2.03	6.24 ±0.27	6.93 ±3.25	17.10 ±9.66	**1.53** ±0.29	1.71 ±0.09	3.36 ±0.57	4.22 ±0.79
CAVIA	57.25 ±12.09	66.25 ±14.73	67.50 ±15.67	68.50 ±16.42	36.72 ±8.83	6.01 ±0.88	3.99 ±0.50	3.27 ±0.24	1.99 ±0.11	1.27 ±0.01	1.21 ±0.00	1.17 ±0.00
TAM (Comp)	63.00 ±5.35	**76.50** ±4.65	**82.75** ±3.86	88.5 ±2.65	**6.15** ±0.91	**3.43** ±0.05	**2.69** ±0.04	**2.13** ±0.02	2.33 ±0.18	1.30 ±0.02	1.23 ±0.01	1.19 ±0.01
TAM (Non-comp)	45.25 ±3.59	72.5 ±3.70	81.5 ±2.89	**89.75** ±0.96	7.80 ±0.09	5.08 ±0.24	3.60 ±0.15	2.42 ±0.08	4.37 ±3.59	**1.27** ±0.01	**1.17** ±0.00	**1.11** ±0.00

Table 5.2: Compositional models for few-shot sequence classification and sequence transduction. All models (except non-compositional TAM) get information on the primitives present in the tasks via extra tokens appended to the input sequence, except that one such primitive is unseen at test time. Non-compositional TAM is not given information about primitives, and estimates a single task embedding instead.

primitives that make up a task. Similar to the previous settings, we hold out sets of values for each of these points and construct the train/test tasks in an analogous manner.

Training For all the models, a sequence of primitive ids representing the primitives that make up the task is appended to the input sequence. These primitive embeddings θ_e are learned along with the other model parameters. To simulate the testing conditions, at training time we pretend some primitives are unknown. For the multitask, matching network and SNAIL baselines we learn an *unknown primitive embedding*, which is used to initialize embeddings of unknown primitives encountered at test time. Although the tasks themselves are harder (because entire primitives are unseen), modeling them is easier because the primitive information is given to the model. For CAVIA and TAM, we infer embeddings for unknown primitives on the fly using gradient descent during train and test. We use 5000 training tasks, and 100 validation and test tasks each. Each training task has 500 examples.

Results Table 5.2 summarizes the results in the compositional setting. We observe similar trends as before for the non-compositional case. Multitask learning becomes competitive only for larger values of k. Vice versa, matching networks and SNAIL suffer with long sequences (larger values of k). TAM performs at least comparably if not better than methods relying on second order derivatives like CAVIA. In fact, CAVIA sometimes fail to converge as shown by the rather large error bars. Finally, the compositional version of TAM often yields higher accuracy than the corresponding non-compositional version, showing that the

Model	Sequence Classification (Acc)			
	1-shot	5-shot	10-shot	20-shot
Input token	0.41	**0.76**	**0.89**	**0.94**
Adapters	**0.43**	0.75	0.88	**0.94**
LayerNorm	0.42	0.64	0.78	0.88

Table 5.3: k-shot classification accuracy when plugging the task embedding in various ways for different values of k.

Arch	Training	Sequence Classification (Acc.)			
		1-shot	5-shot	10-shot	20-shot
LSTM	Multi	0.38	0.57	0.72	0.87
	Alt	0.35	**0.78**	0.83	0.85
Transf	Multi	0.39	0.68	0.80	0.88
	Alt	**0.41**	0.76	**0.89**	**0.94**

Table 5.4: k-shot accuracy for different architectures with multitask and the proposed training algorithms.

model is able to cleverly leverage the additional knowledge about a subset of primitives (two out of three) that compose the new task. Compositionality is particularly helpful with fewer shots (e.g., 1-shot) – with sufficient training examples (e.g., 20-shot) models benefit less from compositionality.

5.5.5 Ablation experiments

Where To Plug Task Embeddings The experiments in the paper so far consider a simple conditioning scheme where the task embedding appears as an additional embedding in the input sequence of token embeddings. We compare this against other ways of incorporating task-specific parameters into the model.

Following Houlsby et al. (2019), we insert adapter layers in the transformer and consider the parameters in the adapter layers as the task embedding. An adapter layer down-projects a representation to a small dimensionality, applies a non-linearity, and up-projects the representation back to the original size. Such layers are added after the self-attention and feed-forward layers in the transformer as residual layers. Houlsby et al. (2019) add adapter layers to a pre-trained BERT model and show that training just the adapter layers works well on downstream tasks. Here the adapter layers are trained using the proposed iterative algorithm, and fine-tuned on test tasks.

Another popular method for adapting pre-trained networks to new tasks is adapting parameters in normalization layers (Perez et al., 2018; Ghiasi et al., 2017). We also consider the scale and bias parameters in the Layer Normalization layers of the transformer as the task embedding.

The results in Table 5.3 on non-compositional classification tasks show that using adapter layer parameters as task embedding yields similar results to the simplest conditioning scheme where the task embedding is fed as an additional input. Using normalization parameters as the task embedding instead performs slightly worse. This shows that our input conditioning scheme is simple yet effective.

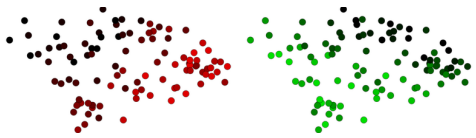


Figure 5.1: 2D PCA projections of task embeddings learned by our algorithm for the gridworld domain. Tasks visualized here have the same start position (4,4). Points are color coded based on horizontal (left plot) and vertical (right plot) coordinates of the end positional corresponding to each task.

Model	Classification Acc., Time	Transduction Ppl., Time	Path-finding Ppl., Time
Multitask	67.4, **30min**	7.2, **23min**	2.9, **20min**
CAVIA	74.8, 3h	4.5, 5.3h	1.5, 3.7h
TAM	**75.0**, 2h	**1.5**, 2.3h	**1.3**, 2.7h

Table 5.5: Training efficiency: Time taken by each training algorithm to reach the best model (identified using validation tasks) and corresponding model performance (non-compositional setting). Performance and time are averaged across $k \in \{1, 5, 10, 20\}$ shots.

Importance of Transformer Architecture The experiments presented in this paper so far have used a transformer architecture. Although transformers are a natural choice for problems involving sequences owing to their recent success, the proposed training algorithm applies equally well to other architectures. We study the impact of swapping out the transformer with a recurrent model in Table 5.4 on non-compositional tasks. We use a bidirectional LSTM with a comparable number of parameters to our transformer model. The classifier head acts on the final representation of the final layer of the LSTM. We examine the performance of the two architectures when trained using both multitasking and the proposed alternating minimization training algorithm. First, we observe that the transformer generally performs better than the recurrent model. Second, the proposed training algorithm yields consistent improvements over the multitask baseline for the transformer. This shows that the proposed algorithm is general, but particularly effective when used in conjunction with the transformer architecture.

Visualizing learned task embeddings In Figure 5.1 we visualize task embeddings learned by the non-compositional TAM model in our gridworld task. We visualize the first two principal components of task embeddings corresponding to tasks which have the same start position. The projections are color coded by the horizontal and vertical coordinates of the end position for each task. This shows that the task embeddings have learned the structure of the tasks.

5.5.6 Discussion

Optimizing Task Embeddings We observed that both CAVIA and TAM generally attain better performance when trained with a larger number of inner loop updates. In this work, we use a maximum of 25 inner loop updates for TAM because it strikes a good balance between finding an optimal task embedding and containing training time. CAVIA performed best with 10 inner loop updates, beyond which we hit the computational limitations of our

hardware. We also found that TAM works better when trained with a number of examples per task much greater than k, in our case 300. All these empirical findings suggest that optimizing for the task embedding and replacing the second order optimization with TAM's first order is an intrinsically difficult problem that requires more iterations and a larger number of examples.

Training Efficiency We discuss the training efficiency of different models in Table 5.5. The multitask baseline is not expensive to train, but it doesn't perform well on few-shot scenarios. CAVIA does well especially in the extreme few-shot scenarios, but has stability issues. TAM is simple, easy to implement, performs comparably or better than the baselines and trains more efficiently than CAVIA.

First vs. Second Order Gradients Double backprop to optimize the test optimization has become a standard method of meta-learning. In the appendix of Finn et al. (2017), and in Zintgraf et al. (2019) (the "first order" variant), similar approaches to TAM were shown to perform relatively worse than the methods with second order gradients.

In contrast, in our settings, we have found that first order gradients (via alternating minimization) are sufficient if done correctly, despite being simpler and more efficient. Although Zintgraf et al. (2019) sometimes outperforms TAM, especially for very small numbers of test examples, TAM is always competitive; with more test examples, TAM is usually superior. TAM always outperforms Finn et al. (2017).

5.6 Conclusion

In this work we have shown how task embeddings naturally fit with Transformers for few-shot learning. In our settings, this approach yields comparable or superior performance to approaches relying on second order derivatives while being computationally more efficient. While we considered synthetic tasks in this work, we expect the proposed idea to be applicable to real-world tasks and large pre-trained transformer models, which can be explored in future work.

This chapter presented a new optimization algorithm to adapt transformer models to the few-shot setting. The remaining chapters explore a different paradigm for learning from limited supervision, where models read a text description in order to learn a new task or adapt models to a new domain. We show that this learning paradigm, when combined with powerful pre-trained text representations, enables zero-shot generalization to new tasks and new domains.

CHAPTER VI

Zero-shot Entity Linking by Reading Entity Descriptions

Traditional methods for Entity Linking focus on linking entities to general databases such as Wikipedia. However, there is a practical need for linking entities in specialized domains such as the internal documents of a company or the characters of a novel. We present the *zero-shot entity linking* task, where mentions must be linked to unseen entities without in-domain labeled data. The goal is to enable robust transfer to highly specialized domains, and so no metadata or alias tables are assumed. In this setting, entities are only identified by text descriptions, and models must rely strictly on language understanding to resolve the new entities. First, we show that strong reading comprehension models pre-trained on large unlabeled data can be used to generalize to unseen entities. Second, we propose a simple and effective adaptive pre-training strategy, which we term *domain-adaptive pre-training* (DAP), to address the domain shift problem associated with linking unseen entities in a new domain. We present experiments on a new dataset that we construct for this task and show that DAP improves over strong pre-training baselines, including BERT. The data and code are available at https://github.com/lajanugen/zeshel.[1]

6.1 Introduction

Entity linking systems have achieved high performance in settings where a large set of disambiguated mentions of entities in a target entity dictionary is available for training. Such systems typically use powerful resources such as a high-coverage alias table, structured data, and linking frequency statistics. For example, Milne and Witten (2008) show that by only using the prior probability gathered from hyperlink statistics on Wikipedia training articles, one can achieve 90% accuracy on the task of predicting links in Wikipedia test articles.

[1] zeshel stands for **ze**ro-**sh**ot **e**ntity **l**inking.

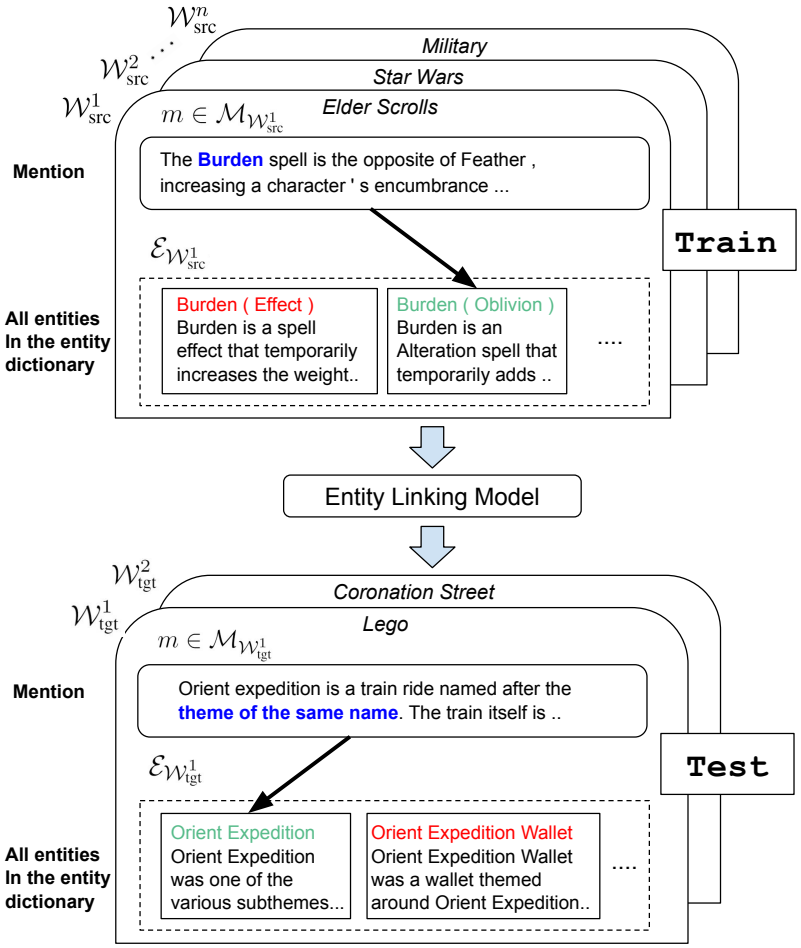


Figure 6.1: **Zero-shot entity linking.** Multiple training and test domains (worlds) are shown. The task has two key properties: (1) It is **zero-shot**, as no mentions have been observed for any of the test world entities during training. (2) Only **textual** (non-structured) information is available.

While most prior works focus on linking to general entity databases, it is often desirable to link to specialized entity dictionaries such as legal cases, company project descriptions, the set of characters in a novel, or a terminology glossary. Unfortunately, labeled data are not readily available and are often expensive to obtain for these specialized entity dictionaries. Therefore, we need to develop entity linking systems that can generalize to unseen specialized entities. Without frequency statistics and meta-data, the task becomes substantially more challenging. Some prior works have pointed out the importance of building entity linking systems that can generalize to unseen entity sets (Sil et al., 2012; Wang et al., 2015), but adopt an additional set of assumptions.

In this work, we propose a new *zero-shot entity linking* task, and construct a new dataset

for it.[2] The target dictionary is simply defined as a set of entities, each with a text description (from a canonical entity page, for example). We do not constrain mentions to named entities, unlike some prior work, which makes the task harder due to large number of candidate entities. In our dataset, multiple entity dictionaries are available for training, with task performance measured on a disjoint set of test entity dictionaries for which no labeled data is available. Figure 6.1 illustrates the task setup. We construct the dataset using multiple sub-domains in Wikia and automatically extract labeled mentions using hyper-links.

Zero-shot entity linking poses two challenges for entity linking models. First, without the availability of powerful alias tables or frequency priors, models must read entity descriptions and reason about the correspondence with the mention in context. We show that a strong reading comprehension model is crucial. Second, since labeled mentions for test entities are not available, models must adapt to new mention contexts and entity descriptions. We focus on both of these challenges.

The contributions of this paper are as follows:

- We propose a new *zero-shot entity linking* task that aims to challenge the generalization ability of entity linking systems with minimal assumptions. We construct a dataset for this task, which will be made publicly available.
- We build a strong baseline by using state-of-the-art reading comprehension models. We show that attention between mention in context and entity descriptions, which has not been used in prior entity linking work, is critical for this task.
- We propose a simple yet novel adaptation strategy called domain-adaptive pre-training (DAP) and show that it can further improve entity linking performance.

6.2 Zero-shot Entity Linking

We first review standard entity linking task definitions and discuss assumptions made by prior systems. We then define the zero-shot entity linking task and discuss its relationship to prior work.

6.2.1 Review: Entity linking

Entity linking (EL) is the task of grounding entity mentions by linking them to entries in a given database or dictionary of entities. Formally, given a mention m and its context, an entity linking system links m to the corresponding entity in an **entity set** $\mathcal{E} = \{e_i\}_{i=1,\ldots,K}$, where K is the number of entities. The standard definition of EL (Bunescu and Pasca, 2006; Roth et al., 2014; Sil et al., 2018) assumes that mention boundaries are provided by users or

[2] Existing datasets are either unsuitable or would have to be artificially partitioned to construct a dataset for this task.

Task	In-Domain	Seen Entity Set	Small Candidate Set	Statistics	Structured Data	Entity dictionary
Standard EL	✓	✓		✓	✓	✓
Cross-Domain EL		✓		✓	✓	✓
Linking to Any DB			✓		✓	✓
Zero-Shot EL						✓

Table 6.1: Assumptions and resources for entity linking task definitions. We classify task definitions based on whether (i) the system is tested on mentions from the training domain (In-Domain), (ii) linked mentions from the target entity set are seen during training (Seen Entity Set), (iii) a small high-coverage candidate set can be derived using alias tables or strict token overlap constraints (Small Candidate Set) and the availability of (iv) Frequency statistics, (v) Structured Data, and (vi) textual descriptions (Entity dictionary).

a mention detection system. The entity set $\mathcal{E}$ can contain tens of thousands or even millions of entities, making this a challenging task. In practice, many entity linking systems rely on the following resources or assumptions:

Single entity set This assumes that there is a single comprehensive set of entities $\mathcal{E}$ shared between training and test examples.

Alias table An alias table contains entity candidates for a given mention string and limits the possibilities to a relatively small set. Such tables are often compiled from a labeled training set and domain-specific heuristics.

Frequency statistics Many systems use frequency statistics obtained from a large labeled corpus to estimate entity popularity and the probability of a mention string linking to an entity. These statistics are very powerful when available.

Structured data Some systems assume access to structured data such as relationship tuples (e.g., (*Barack Obama*, *Spouse*, *Michelle Obama*)) or a type hierarchy to aid disambiguation.

6.2.2 Task Definition

The main motivation for this task is to expand the scope of entity linking systems and make them generalizable to unseen entity sets for which none of the powerful resources listed above are readily available. Therefore, we drop the above assumptions and make one weak assumption: the existence of an **entity dictionary** $\mathcal{E} = \{(e_i, d_i)\}_{i=1,..,K}$, where d_i is a text description of entity e_i.

Our goal is to build entity linking systems that can generalize to new domains and entity dictionaries, which we term *worlds*. We define a world as $\mathcal{W} = (\mathcal{M}_\mathcal{W}, \mathcal{U}_\mathcal{W}, \mathcal{E}_\mathcal{W})$, where

$\mathcal{M}_\mathcal{W}$ and $\mathcal{U}_\mathcal{W}$ are distributions over mentions and documents from the world, respectively, and $\mathcal{E}_\mathcal{W}$ is an entity dictionary associated with $\mathcal{W}$. Mentions m from $\mathcal{M}_\mathcal{W}$ are defined as mention spans in documents from $\mathcal{U}_\mathcal{W}$. We assume the availability of labelled mention, entity pairs from one or more source worlds $\mathcal{W}^1_{\text{src}} \dots \mathcal{W}^n_{\text{src}}$ for training. At test time we need to be able to label mentions in a new world $\mathcal{W}_{\text{tgt}}$. Note that the entity sets $\mathcal{E}_{\mathcal{W}^1_{\text{src}}}, \dots, \mathcal{E}_{\mathcal{W}^n_{\text{src}}}, \mathcal{E}_{\mathcal{W}_{\text{tgt}}}$ are disjoint. See Figure 6.1 for an illustration of several training and test worlds.

We additionally assume that samples from the document distribution $\mathcal{U}_{\mathcal{W}_{\text{tgt}}}$ and the entity descriptions $\mathcal{E}_{\mathcal{W}_{\text{tgt}}}$ are available for training. These samples can be used for unsupervised adaptation to the target world. During training, mention boundaries for mentions in $\mathcal{W}_{\text{tgt}}$ are not available. At test time, mention boundaries are provided as input.

6.2.3 Relationship to other EL tasks

We summarize the relationship between the newly introduced zero-shot entity linking task and prior EL task definitions in Table 6.1.

Standard EL While there are numerous differences between EL datasets (Bunescu and Pasca, 2006; Ling et al., 2015), most focus on a standard setting where mentions from a comprehensive test entity dictionary (often Wikipedia) are seen during training, and rich statistics and meta-data can be utilized (Roth et al., 2014). Labeled in-domain documents with mentions are also assumed to be available.

Cross-Domain EL Recent work has also generalized to a cross-domain setting, linking entity mentions in different types of text, such as blogposts and news articles to the Wikipedia KB, while only using labeled mentions in Wikipedia for training (e.g., Gupta et al. (2017a); Le and Titov (2018), *inter alia*).

Linking to Any DB Sil et al. (2012) proposed a task setup very similar to ours, and later work (Wang et al., 2015) has followed a similar setting. The main difference between zero-shot EL and these works is that they assumed either a high-coverage alias table or high-precision token overlap heuristics to reduce the size of the entity candidate set (i.e., to less than four in Sil et al. (2012)) and relied on structured data to help disambiguation. By compiling and releasing a multi-world dataset focused on learning from textual information, we hope to help drive progress in linking entities for a broader set of applications.

Work on word sense disambiguation based on dictionary definitions of words is related as well (Chaplot and Salakhutdinov, 2018), but this task exhibits lower ambiguity and existing formulations have not focused on domain generalization.

6.3 Dataset Construction

We construct a new dataset to study the zero-shot entity linking problem using documents from Wikia.[3] Wikias are community-written encyclopedias, each specializing in a particular subject or theme such as a fictional universe from a book or film series. Wikias have many interesting properties suitable for our task. Labeled mentions can be automatically extracted based on hyperlinks. Mentions and entities have rich document context that can be exploited by reading comprehension approaches. Each Wikia has a large number of unique entities relevant to a specific theme, making it a useful benchmark for evaluating domain generalization of entity linking systems.

We use data from 16 Wikias, and use 8 of them for training and 4 each for validation and testing. To construct data for training and evaluation, we first extract a large number of mentions from the Wikias. Many of these mentions can be easily linked by string matching between mention string and the title of entity documents. These mentions are downsampled during dataset construction, and occupy a small percentage (5%) of the final dataset. While not completely representative of the natural distribution of mentions, this data construction method follows recent work that focuses on evaluating performance on the challenging aspects of the entity linking problem (e.g., Gupta et al. (2017a) selected mentions with multiple possible entity candidates for assessing in-domain unseen entity performance). Each Wikia document corresponds to an entity, represented by the title and contents of the document. These entities, paired with their text descriptions, comprise the entity dictionary.

Since the task is already quite challenging, we assume that the target entity exists in the entity dictionary and leave NIL recognition or clustering (NIL mentions/entities refer to entities non-existent in the knowledge-base) to future editions of the task and dataset.

We categorize the mentions based on token overlap between mentions and the corresponding entity title as follows. *High Overlap*: title is identical to mention text, *Multiple Categories*: title is mention text followed by a disambiguation phrase (e.g., mention string: 'Batman', title: 'Batman (Lego)'), *Ambiguous substring*: mention is a substring of title (e.g., mention string: 'Agent', title: 'The Agent'). All other mentions are categorized as *Low Overlap*. These mentions respectively constitute approximately 5%, 28%, 8% and 59% of the mentions in the dataset.

Table 6.2 shows some statistics of the dataset. Each domain has a large number of entities ranging from 10,000 to 100,000. The training set has 49,275 labeled mentions. To examine the in-domain generalization performance, we construct heldout sets *seen* and *unseen* of 5,000 mentions each, composed of mentions that link to only entities that were seen or

[3] https://www.wikia.com.

Split	World	Entities	Mentions		
			Train	Evaluation	
				Seen	Unseen
Training	American Football	31929	3898	410	333
	Doctor Who	40281	8334	819	702
	Fallout	16992	3286	337	256
	Final Fantasy	14044	6041	629	527
	Military	104520	13063	1356	1408
	Pro Wrestling	10133	1392	151	111
	StarWars	87056	11824	1143	1563
	World of Warcraft	27677	1437	155	100
Validation	Coronation Street	17809	0	0	1464
	Muppets	21344	0	0	2028
	Ice Hockey	28684	0	0	2233
	Elder Scrolls	21712	0	0	4275
Test	Forgotten Realms	15603	0	0	1200
	Lego	10076	0	0	1199
	Star Trek	34430	0	0	4227
	YuGiOh	10031	0	0	3374

Table 6.2: Zero-shot entity linking dataset based on Wikia.

unseen during training, respectively. The validation and test sets have 10,000 mentions each (all of which are unseen).

Table 6.3 shows examples of mentions and entities in the dataset. The vocabulary and language used in mentions and entity descriptions differs drastically between the different domains. In addition to acquiring domain specific knowledge, understanding entity descriptions and performing reasoning is required in order to resolve mentions.

6.4 Models for Entity Linking

We adopt a two-stage pipeline consisting of a fast candidate generation stage, followed by a more expensive but powerful candidate ranking stage.

6.4.1 Candidate generation

Without alias tables for standard entity linking, a natural substitute is to use an IR approach for candidate generation. We use BM25, a variant of TF-IDF to measure similarity between mention string and candidate documents.[4] Top-k entities retrieved by BM25 scoring with

[4]We also experimented with using the mention+context text but this variant performs substantially worse.

Coronation Street		
Mention	She told ray that Dickie and Audrey had met up again and tried to give their marriage another go . . . I don't want to see **her** face again . . . "	
✓	Dickie Fleming	Richard "Dickie" Fleming lived in coronation street with his wife Audrey from 1968 to 1970.
	Audrey Fleming	Audrey Fleming (neé bright) was a resident of 3 coronation street from 1968 to 1970 . Audrey married Dickie Fleming . . .
	Zeedan Nazir	Zeedan Nazir is the son of the Late Kal and Jamila Nazir . . .
Star Wars		
Mention	The droid acted as Moff Kilran's representative on board the Black Talon, an **Imperial transport ship**.	
✓	Gage-class transport	The Gage-class transport was a transport design used by the reconstituted Sith Empire of the Great Galactic War.
	Imperial Armored Transport	The Kuat Drive Yards Imperial Armored Transport was fifty meters long and carried ten crewmen and twenty soldiers.
	M-class Imperial Attack Transport	The M-class Imperial Attack Transport was a type of starship which saw service in the Imperial Military during the Galactic War.

Table 6.3: Example mention and entity candidates from Coronation Street and Star Wars. Note that the language usage is very different across different Worlds.

Lucene[5] are used for training and evaluation. In our experiments k is set to 64. The coverage of the top-64 candidates is less than 77% on average, indicating the difficulty of the task and leaving substantial room for improvement in the candidate generation phase.

6.4.2 Candidate ranking

Since comparing two texts—a mention in context and a candidate entity description—is a task similar to reading comprehension and natural language inference tasks, we use an architecture based on a deep Transformer (Vaswani et al., 2017b) which has achieved state-of-the-art performance on such tasks (Radford et al., 2018; Devlin et al., 2019).

As in BERT (Devlin et al., 2019), the mention in context m and candidate entity description e, each represented by 128 word-piece tokens, are concatenated and input to the model as a sequence pair together with special start and separator tokens: ([CLS] m [SEP] e [SEP]). Mention words are signaled by a special embedding vector that is added to the mention word embeddings. The Transformer encoder produces a vector representation $h_{m,e}$ of the input pair, which is the output of the last hidden layer at the special pooling token [CLS]. Entities in a given candidate set are scored as $w^\top h_{m,e}$ where w is a learned parameter vector, and the model is trained using a softmax loss. An architecture with 12 layers, hidden dimension size 768 and 12 attention heads was used in our experiments. We refer to this model as

[5] http://lucene.apache.org/

Full-Transformer. By jointly encoding the entity description and the mention in context with a Transformer, they can attend to each other at every layer.

Note that prior neural approaches for entity linking have not explored such architectures with deep cross-attention. To assess the value of this departure from prior work, we implement the following two variants: (*i*) **Pool-Transformer**: a siamese-like network which uses two deep Transformers to separately derive single-vector representations of the mention in context, h_m, and the candidate entity, h_e; they take as input the mention in context and entity description respectively, together with special tokens indicating the boundaries of the texts: ([CLS] m [SEP]) and ([CLS] e [SEP]), and output the last hidden layer encoding at the special start token. The scoring function is $h_m^\top h_e$. Single vector representations for the two components have been used in many prior works, e.g., Gupta et al. (2017a). (*ii*) **Cand-Pool-Transformer**: a variant which uses single vector entity representations but can attend to individual tokens of the mention and its context as in Ganea and Hofmann (2017). This architecture also uses two Transformer encoders, but introduces an additional attention module which allows h_e to attend to individual token representations of the mention in context.

In the experiments section, we also compare to re-implementations of Gupta et al. (2017a) and Ganea and Hofmann (2017), which are similar to Pool-Transformer and Cand-Pool-Transformer respectively but with different neural architectures for encoding.

6.5 Adapting to the Target World

We focus on using unsupervised pre-training to ensure that downstream models are robust to target domain data. There exist two general strategies for pre-training: (1) task-adaptive pre-training, and (2) open-corpus pre-training. We describe these below, and also propose a new strategy: domain-adaptive pre-training (DAP), which is complementary to the two existing approaches.

Task-adaptive pre-training Glorot et al. (2011); Chen et al. (2012); Yang and Eisenstein (2015), *inter alia*, pre-trained on the source and target domain unlabeled data jointly with the goal of discovering features that generalize across domains. After pre-training, the model is fine-tuned on the source-domain labeled data.[6]

Open-corpus pre-training Instead of explicitly adapting to a target domain, this approach simply applies unsupervised pre-training to large corpora before fine-tuning on the source-domain labeled data. Examples of this approach include ELMo (Peters et al., 2018a), OpenAI GPT (Radford et al., 2018), and BERT (Devlin et al., 2019). Intuitively, the target-domain

[6]In many works, the learned representations are kept fixed and only higher layers are updated.

distribution is likely to be partially captured by pre-training if the open corpus is sufficiently large and diverse. Indeed, open-corpus pre-training has been shown to benefit out-of-domain performance far more than in-domain performance (He et al., 2018).

Domain-adaptive pre-training In addition to pre-training stages from other approaches, we propose to insert a penultimate *domain adaptive pre-training* (DAP) stage, where the model is pre-trained *only* on the target-domain data. As usual, DAP is followed by a final fine-tuning stage on the source-domain labeled data. The intuition for DAP is that representational capacity is limited, so models should prioritize the quality of target domain representations above all else.

We introduce notation to describe various ways in which pre-training stages can be composed.

- U_{src} denotes text segments from the union of source world document distributions $\mathcal{U}_{\mathcal{W}^1_{\text{src}}} \ldots \mathcal{U}_{\mathcal{W}^n_{\text{src}}}$.
- U_{tgt} denotes text segments from the document distribution of a target world $\mathcal{W}_{\text{tgt}}$.
- $U_{\text{src+tgt}}$ denotes randomly interleaved text segments from both U_{src} and U_{tgt}.
- U_{WB} denotes text segments from open corpora, which in our experiments are Wikipedia and the BookCorpus datasets used in BERT.

We can chain together a series of pre-training stages. For example, $U_{\text{WB}} \rightarrow U_{\text{src+tgt}} \rightarrow U_{\text{tgt}}$ indicates that the model is first pre-trained on the open corpus, then pre-trained on the combined source and target domains, then pre-trained on only the target domain, and finally fine-tuned on the source-domain labeled data.[7] We show that chaining together different pre-training strategies provides additive gains.

6.6 Experiments

Pre-training We use the BERT-Base model architecture in all our experiments. The Masked LM objective (Devlin et al., 2019) is used for unsupervised pre-training. For fine-tuning language models (in the case of multi-stage pre-training) and fine-tuning on the Entity-Linking task, we use a small learning rate of 2e-5, following the recommendations from Devlin et al. (2019). For models trained from scratch we use a learning rate of 1e-4.

[7] We use the notation U_x interchangeably to mean both the unsupervised data x and the strategy to pre-train on x.

Model	Resources	Avg Acc
Edit-distance	$\emptyset$	16.49
TF-IDF (BM25)	$\emptyset$	26.06
Ganea and Hofmann (2017)	GloVe	26.96
Gupta et al. (2017a)	GloVe	27.03
Full-Transformer	$\emptyset$	19.17
Full-Transformer (Pre-trained)	U_{src}	66.55
Full-Transformer (Pre-trained)	U_{tgt}	67.87
Full-Transformer (Pre-trained)	$U_{\text{src+tgt}}$	67.91
Pool-Transformer (Pre-trained)	U_{WB}	57.61
Cand-Pool-Trans. (Pre-trained)	U_{WB}	52.62
Full-Transformer (Pre-trained)	U_{WB}	76.06

Table 6.4: Baseline results for Zero-shot Entity Linking. Averaged normalized Entity-Linking accuracy on all validation domains. $U_{\text{src+tgt}}$ refers to masked language model pre-training on unlabeled data from training and validation worlds.

Evaluation	Accuracy
Training worlds, seen	87.74
Training worlds, unseen	82.96
Validation worlds, unseen	76.06

Table 6.5: Performance of the Full-Transformer (U_{WB}) model evaluated on seen and unseen entities from the training and validation worlds.

Evaluation We define the *normalized* entity-linking performance as the performance evaluated on the subset of test instances for which the gold entity is among the top-k candidates retrieved during candidate generation. The *unnormalized* performance is computed on the entire test set. Our IR-based candidate generation has a top-64 recall of 76% and 68% on the validation and test sets, respectively. The unnormalized performance is thus upper-bounded by these numbers. Strengthening the candidate generation stage improves the unnormalized performance, but this is outside the scope of our work. Average performance across a set of worlds is computed by macro-averaging. Performance is defined as the accuracy of the single-best identified entity (top-1 accuracy).

Baselines We first examine some baselines for zero-shot entity linking in Table 6.4. We include naive baselines such as Levenshtein edit-distance and TF-IDF, which compare the mention string against candidate entity title and full document description, respectively, to rank candidate entities.

We re-implemented recent neural models designed for entity linking (Ganea and Hof-

mann, 2017; Gupta et al., 2017a), but did not expect them to perform well since the original systems were designed for settings where labeled mentions or meta-data for the target entities were available. The poor performance of these models validates the necessity of using strong reading comprehension models for zero-shot entity linking.

When using the Full-Transformer model, pre-training is necessary to achieve reasonable performance. We present results for models pre-trained on different subsets of our task corpus (U_{src}, U_{tgt}, $U_{\text{src+tgt}}$) as well as pre-training on an external large corpus (U_{WB}). We observe that the choice of data used for pre-training is important.

In Table 6.4 we also compare the Pool-Transformer, Candidate-Pool-Transformer and Full-Transformer. The significant gap between Full-Transformer and the other variants shows the importance of allowing fine-grained comparisons between the two inputs via the cross attention mechanism embedded in the Transformer. We hypothesize that prior entity linking systems did not need such powerful reading comprehension models due to the availability of strong additional meta information. The remaining experiments in the paper use the Full-Transformer model, unless mentioned otherwise.

6.6.1 Generalization to Unseen Entities and New Worlds

To analyze the impact of unseen entities and domain shift in zero-shot entity linking, we evaluate performance on a more standard in-domain entity linking setting by making predictions on held out mentions from the training worlds. Table 6.5 compares entity linking performance for different entity splits. Seen entities from the training worlds are unsurprisingly the easiest to link to. For unseen entities from the training world, we observe a 5-point drop in performance. Entities from new worlds (which are by definition unseen and are mentioned in out-of-domain text) prove to be the most difficult. Due to the shift in both the language distribution and entity sets, we observe a 11-point drop in performance. This large generalization gap demonstrates the importance of adaptation to new worlds.

6.6.2 Impact of Domain Adaptive Pre-training

Our experiments demonstrate that DAP improves on three state-of-the-art pre-training strategies:

- $U_{\textbf{src+tgt}}$: task-adaptive pre-training, which combines source and target data for pre-training (Glorot et al., 2011).[8]
- $U_{\textbf{WB}}$: open-corpus pre-training, which uses Wikipedia and the BookCorpus for pre-training (We use a pre-trained BERT model (Devlin et al., 2019)).

[8] We use Masked LM and Transformer encoder, which are more powerful than the instantiation in Glorot et al. (2011).

Pretraining	$\mathcal{W}^1_{\text{tgt}}$	$\mathcal{W}^2_{\text{tgt}}$	$\mathcal{W}^3_{\text{tgt}}$	$\mathcal{W}^4_{\text{tgt}}$	Avg
$U_{\text{src+tgt}}$ Glorot et al. (2011)†	73.19	71.61	62.16	64.69	67.91
$U_{\text{src+tgt}} \rightarrow U_{\text{tgt}}$ (DAP)	79.20	75.55	66.85	66.72	**72.08**
U_{WB} Devlin et al. (2019)	83.40	79.00	73.03	68.82	76.06
$U_{\text{WB}} \rightarrow U_{\text{tgt}}$ (DAP)	81.68	81.34	73.17	71.97	**77.04**
$U_{\text{WB}} \rightarrow U_{\text{src+tgt}}$	82.92	79.00	72.62	69.55	76.02
$U_{\text{WB}} \rightarrow U_{\text{src+tgt}} \rightarrow U_{\text{tgt}}$ (DAP)	82.82	81.59	75.34	72.52	**78.07**

Table 6.6: Impact of using Domain Adaptive Pre-training. We fine-tune all the models on the source labeled data after pretraining. **Note:** *src* represents the union of all 8 training worlds and we adapt to one *tgt* world at a time. The target worlds are $\mathcal{W}^1_{\text{tgt}}$: *Coronation street*, $\mathcal{W}^2_{\text{tgt}}$: *Muppets*, $\mathcal{W}^3_{\text{tgt}}$: *Ice hockey*, $\mathcal{W}^4_{\text{tgt}}$: *Elder scrolls*. †We refer to Glorot et al. (2011) for the idea of training a denoising autoencoder on source and target data together rather than the actual implementation. See text for more details.

- $U_{\textbf{WB}} \rightarrow U_{\textbf{src+tgt}}$: the previous two strategies chained together. While no prior work has applied this approach to domain adaptation, a similar approach for task adaptation was proposed by Howard and Ruder (2018).

The results are in Table 6.6. DAP improves all pre-training strategies with an additional pre-training stage on only target-domain data. The best setting, $U_{\text{WB}} \rightarrow U_{\text{src+tgt}} \rightarrow U_{\text{tgt}}$, chains together all existing strategies. DAP improves the performance over a strong pre-trained model (Devlin et al., 2019) by 2%.

To further analyze the results of DAP, we plot the relationships between the accuracy of Masked LM (MLM accuracy) on target unlabeled data and the final target normalized accuracy (after fine-tuning on the source labeled data) in Figure 6.2. Adding an additional pre-training stage on the target unlabeled data unsurprisingly improves the MLM accuracy. More interestingly, we find that improvements in MLM accuracy are consistently followed by improvements in entity linking accuracy. It is intuitive that performance on unsupervised objectives reflect the quality of learned representations and correlate well with downstream performance. We show empirically that this trend holds for a variety of pre-training strategies.

6.6.3 Test results and performance analysis

Table 6.7 shows the normalized and unnormalized Entity Linking performance on test worlds. Our best model that chains together all pre-training strategies achieves normalized accuracy of 77.05% and unnormalized accuracy of 56.58%. Note that the unnormalized accuracy corresponds to identifying the correct entity from tens of thousands of candidate entities.

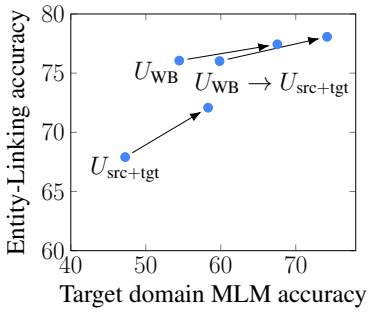


Figure 6.2: Relationship between MLM accuracy of pre-trained model and Entity-Linking performance of the fine-tuned model, evaluated on target domains.

Pre-training	EL Accuracy	
	N. Acc.	U. Acc.
U_{WB} Devlin et al. (2019)	75.06	55.08
$U_{\text{WB}} \rightarrow U_{\text{tgt}}$ (DAP)	76.17	55.88
$U_{\text{WB}} \rightarrow U_{\text{src+tgt}} \rightarrow U_{\text{tgt}}$ (DAP)	**77.05**	**56.58**

Table 6.7: Performance on test domains with Full-Transformer. **N. Acc** represents the normalized accuracy. **U. Acc** represents the unnormalized accuracy. The unnormalized accuracy is upper-bounded by 68%, the top-64 recall of the candidate generation stage.

Mention Category	Recall@64	EL Accuracy	
		N. Acc.	U. Acc.
High Overlap	99.28	87.64	87.00
Ambiguous Substring	88.03	75.89	66.81
Multiple categories	84.88	77.27	65.59
Low Overlap	54.37	71.46	38.85

Table 6.8: Performance on test domains categorized by mention categories. Recall@64 indicates top-64 performance of candidate generation. N. Acc. and U. Acc. are respectively the normalized and unnormalized accuracies.

To analyze the mistakes made by the model, we compare EL accuracy across different mention categories in Table 6.8. Candidate generation (Recall@64) is poor in the Low Overlap category. However, the ranking model performs in par with other hard categories for these mentions. Overall EL accuracy can thus be improved significantly by strengthening candidate generation.

6.7 Related Work

We discussed prior entity linking task definitions and compared them to our task in section 6.2. Here, we briefly overview related entity linking models and unsupervised domain adaptation methods.

Entity linking models Entity linking given mention boundaries as input can be broken into the tasks of candidate generation and candidate ranking. When frequency information or alias tables are unavailable, prior work has used measures of similarity of the mention string to entity names for candidate generation (Sil et al., 2012; Murty et al., 2018). For candidate

ranking, recent work employed distributed representations of mentions in context and entity candidates and neural models to score their compatibility. Mentions in context have been represented using e.g., CNN (Murty et al., 2018), LSTM (Gupta et al., 2017a), or bag-of-word embeddings (Ganea and Hofmann, 2017). Entity descriptions have been represented using similar architectures. To the best of our knowledge, while some models allow for cross-attention between single-vector entity embeddings and mention-in-context token representations, no prior works have used full cross-attention between mention+context and entity descriptions.

Prior work on entity linking tasks most similar to ours used a linear model comparing a mention in context to an entity description and associated structured data (Sil et al., 2012). Sil et al. (2012) also proposed a distant supervision approach which could use first-pass predictions for mentions in the target domain as noisy supervision for re-training an in-domain model. We believe this approach is complementary to unsupervised representation learning and could bring additional benefits. In another task similar to ours, Wang et al. (2015) used collective inference and target database relations to obtain good performance without (domain, target database)-specific labeled training data. Collective inference is another promising direction, but could have limited success when no metadata is available.

Unsupervised domain adaptation There is a large body of work on methods for unsupervised domain adaptation, where a labeled training set is available for a source domain and unlabeled data is available for the target domain. The majority of work in this direction assume that training and test examples consist of (x, y) pairs, where y is in a fixed shared label set $\mathcal{Y}$. This assumption holds for classification and sequence labeling, but not for zero-shot entity linking, since the source and target domains have disjoint labels.

Most state-of-the-art methods learn non-linear shared representations of source and target domain instances, through denoising training objectives (Eisenstein, 2018). In Section 6.5, we overviewed such work and proposed an improved domain adaptive pre-training method.

Adversarial training methods (Ganin et al., 2016), which have also been applied to tasks where the space $\mathcal{Y}$ is not shared between source and target domains (Cohen et al., 2018), and multi-source domain adaptation methods (Zhao et al., 2018; Guo et al., 2018) are complementary to our work and can contribute to higher performance.

6.8 Conclusion

We introduce a new task for zero-shot entity linking, and construct a multi-world dataset for it. The dataset can be used as a shared benchmark for entity linking research focused on specialized domains where labeled mentions are not available, and entities are defined

through descriptions alone. A strong baseline is proposed by combining powerful neural reading comprehension with domain-adaptive pre-training.

Future variations of the task could incorporate NIL recognition and mention detection (instead of mention boundaries being provided). The candidate generation phase leaves significant room for improvement. We also expect models that jointly resolve mentions in a document would perform better than resolving them in isolation.

In this chapter we showed how models can learn to recognize entities in unknown domains by exploiting text descriptions of those entities. The next chapter applies this principle to learn new tasks where text descriptions of tasks are assumed to be available. We show that intelligent agents can be trained to perform unseen tasks by combining skills they learned during training based on task descriptions.

CHAPTER VII

Learning Zero-shot Compositional Tasks from Language Instructions

Systematic compositionality – the ability to combine learned knowledge and skills to solve novel tasks – is a key aspect of generalization in humans that allows us to understand and perform tasks described by novel language utterances. While progress has been made in supervised learning settings, no work has yet studied compositional generalization of a reinforcement learning agent following natural language instructions in an embodied environment. We develop a set of tasks in a photo-realistic simulated kitchen environment that allow us to study the degree to which a behavioral policy captures the systematicity in language by studying its zero-shot generalization performance on held out natural language instructions. We show that our agent which leverages a novel additive action-value decomposition in tandem with attention-based subgoal prediction is able to exploit composition in text instructions to generalize to unseen tasks.

7.1 Introduction

Human language is characterized by systematic compositionality: one can combine known components – such as words or phrases – to produce novel linguistic combinations (Chomsky, 2009). This is a key aspect of generalization in humans and enables us to understand and perform tasks specified by novel language utterances over familiar words or phrases. If you know what a "laptop" and a "fridge" are, you can easily understand how to perform the task "place the laptop in the fridge" even if you have never placed a laptop in a fridge.

Prior work studying the linguistic "systematicity" of neural networks have focused on sequence mapping tasks in a supervised learning setting (Lake and Baroni, 2018; Lake, 2019; Andreas, 2019). In this work, we are interested in compositional generalization of a reinforcement learning agent following natural language instructions in an embodied

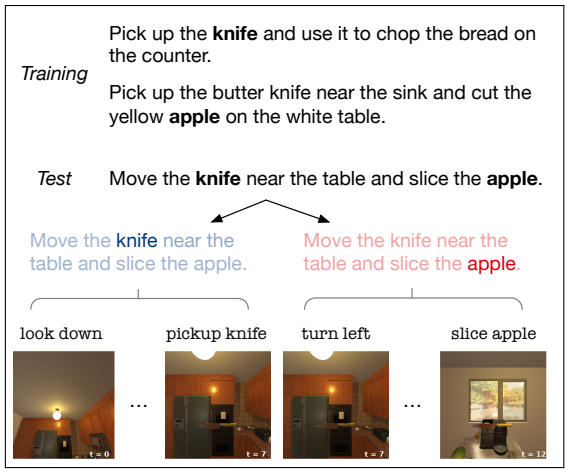


Figure 7.1: Zero-shot generalization to an unseen task of slicing an apple. The test task is composed of known primitive subtasks – *picking up a knife* and *slicing the apple* – each of which were encountered in training tasks. Our agent learns to decompose a natural language task description into subtasks using attention and executes them using low-level actions.

environment. In particular, we explore the hypothesis that a language-conditioned reinforcement learning agent with a compositional inductive bias in its behavioral policy will exhibit systematic generalization to unobserved natural language instructions.

There has been a flurry of recent work on embodied learning tasks such as question answering (Gupta et al., 2017b), navigation (Anderson et al., 2018) and object interaction (Shridhar et al., 2020; Carvalho et al., 2020) in embodied settings. In particular, the ALFRED task (Shridhar et al., 2020) studies agents that exploit detailed natural language instructions to generalize to novel instructions in novel environments at test time. Such existing benchmarks offer limited flexibility to study systematic generalization since (i) the benchmarks were not built for this purpose and it is unclear to what extent systematic generalization skills are required to solve the tasks and (ii) the tasks demand challenging reasoning skills such as visual recognition and planning over large number of time-steps which makes it difficult to study compositional generalization ability in isolation.

In this work we develop a set of tasks in the AI2Thor virtual home environment (Kolve et al., 2017) which test the compositionality of embodied agents. In order to make progress in systematic generalization, we make two simplifying assumptions: we assume access to an oracle object recognizer and we study generalization in a single kitchen layout. This allows us to study the degree to which a policy captures the systematicity in language by studying its zero-shot generalization performance on held out natural language instructions.

Despite these simplifications, agents still need to understand the instruction to figure out the sequence of object interactions that need to be performed and act over many time-steps

with limited guidance. In order to succesfully generalize at test time, an agent needs to learn to ground natural language instructions to temporally extended goal-oriented behaviors or "skills" in a compositional manner to perform novel tasks that are compositions of the tasks presented at train time. We leverage this setting to develop and study a policy with an inductive bias for compositionality and show that this enables systematic generalization in the context of combining behavioral skills learned purely from reward without expert demonstrations.

We present an attention-based agent that learns to predict subgoals from language instructions via a learned attention mechanism. Our agent uses these subgoals with a novel policy parametrization which decomposes the action-value function in an additive fashion that enables estimating the action-value for novel object-interactions composed of objects and interactions experienced during training.

We show evidence that this parametrization facilitates exploiting the compositional nature of text instructions by showing systematic generalization to both unseek task descriptions and unseen tasks. We present an example in Fig. 7.1, where the agent is able to systematically generalize the behavior "pickup up the knife" to "move the knife" and "cut the yellow apple" to "slice the apple". Thanks to the additive inductive bias afforded by our action-value parametrization, it is able to compose these behaviors to perform the novel task "move the knife near the table and slice the apple" at test time.

7.2 Related work

Compositional generalization Prior work has studied compositional generalization in sequence mapping tasks. Benchmarks such as SCAN (Lake and Baroni, 2018) and gSCAN (Ruis et al., 2020) study translating synthetic text descriptions to an action sequence (e.g. jump twice $\rightarrow$ JUMP JUMP). gSCAN couples SCAN instances with entities in a grid environment and solving a task requires grounding the text and entities similar to our work. Prior approaches for these benchmarks impose compositional inductive biases in models by augmenting models with memory (Lake, 2019) and data augmentation (Andreas, 2019). In this work we use attention mechanisms and introduce a novel poilcy parameterization to impose compositional inductive biases.

Text based embodied control Advances in photo-realistic simulation environments such as DeepMind Lab (Beattie et al., 2016) and AI2Thor (Kolve et al., 2017) have driven recent progress in embodied agents that learn from text instructions. Chaplot et al. (2018) consider a simple navigation task where an agent has to move to an object specified by a set of attributes such as shape and color. They propose the gated attention model to generalize compositionally in the attribute space. Hill et al. (2019) consider systematic

generalization in 2D and 3D environments with synthetic text instructions. Compared to these work, we consider object interaction tasks in a photo realistic simulated environment with human-authored language instructions.

ALFRED Shridhar et al. (2020) couples tasks in the AI2Thor environment with detailed text descriptions of tasks. In contrast, we consider a simplified setup of learning compositional skills from high-level task descriptions. We further do not assume access to expert task demonstrations. These assumptions allows us to focus on compositional generalization to zero-shot tasks, which is not the main goal of the ALFRED benchmark. However, the approach presented here can potentially be applicable to ALFRED when combined with learning from demonstrations.

Hierarchical Reinforcement Learning Learning to directly map percepts to low-level action sequences can be challenging. An alternative hierarchical approach is to first come up with a sequence of subtasks, which can be considered as high-level actions (Andreas et al., 2017; Zhu et al., 2017). Each of those subtasks can then be realized using low-level actions. Our policy has an implicit hierarchical structure where latent subgoals are represented as text embeddings using attention. Language was used as an abstraction for the high-level policy in Jiang et al. (2019a) for object rearrangement tasks based on the CLEVR engine (Johnson et al., 2017).

Finally, generalization to unseen instructions has been considered in prior work such as Oh et al. (2017); Lynch and Sermanet (2020), although compositional generalization is not their main focus.

7.3 Problem

We consider an embodied agent acting in a kitchen environment to solve basic tasks from language instructions (See Fig. 7.4 for an example task). At the beginning of an episode the agent receives a text instruction τ. Our goal is to learn a policy $\pi(a|s, \tau); a \in \mathcal{A}, s \in \mathcal{S}$ that predicts actions in order to complete tasks.

The agent state s is partially observable – it receives an egocentric observation *obs* of the scene. We further assume that an oracle object recognition model provides the object ids for objects in the egocentric observation.

The action space consists of navigation and object interaction actions $\mathcal{A} = \mathcal{A}_{\text{nav}} \cup \mathcal{A}_{\text{int}}$. There are 8 navigation actions $\mathcal{A}_{\text{nav}} = \{$*move forward, move back, move left, move right, turn left, turn right, look up, look down*$\}$. Interaction actions $\mathcal{A}_{\text{int}} = \mathcal{B} \times \mathcal{I}$ are specified using an interaction $b \in \mathcal{B}$ and an object id $o \in \mathcal{I}$ where $\mathcal{B} = \{$*pickup, place, slice, toggle on, toggle off, turn on, turn off*$\}$ and $\mathcal{I}$ is a pre-defined set of identifiers of objects that are available to the agent for interaction in the current observation.

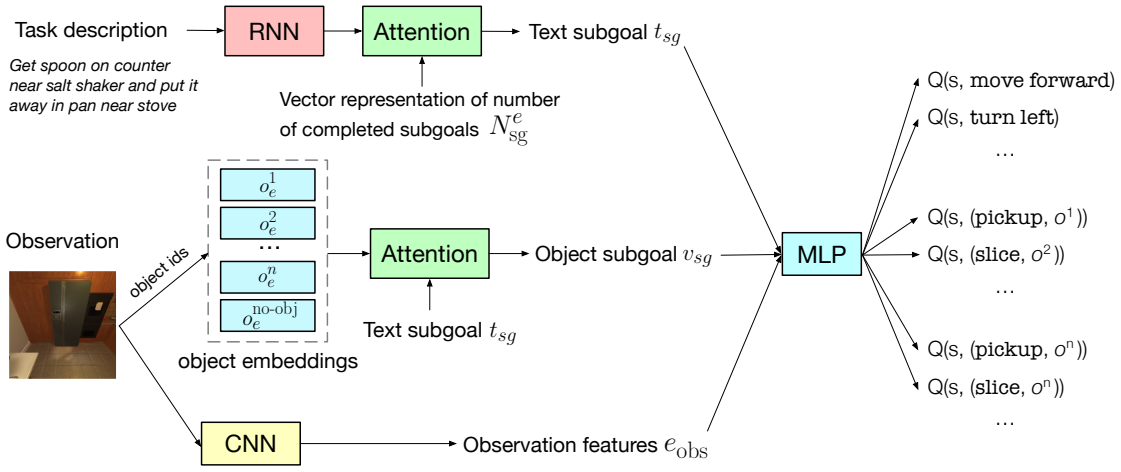


Figure 7.2: Approach Overview: We perform attention over the text instruction to construct an embedding t_{sg} that represents the current subgoal. The text embedding subgoal t_{sg} attends to scene object embeddings to construct an object subgoal representation v_{sg}. An MLP takes t_{sg}, v_{sg} and observation features e_{obs} as input and predicts state-action values $Q(s, a)$. The entire model is trained end-to-end using Q-learning. See text for details.

The agent receives a positive reward for successfully completing a task. It also receives a small negative reward for every time-step. In addition, we also assume that every correct object interaction receives a positive reward. In addition to providing a denser learning signal, the rewards are also used to identify subgoals as described in section 7.4.1. In practice such dense rewards may be unavailable, but this is outside the scope of our study and left as future work.

7.4 Approach

We approach the problem by considering a task τ to be composed of subgoals $g_1, ..., g_n$, where each subgoal g_i involves navigating to a particular object an interacting with it. For example, the task *place an apple on the table* involves finding the apple and picking it up, followed by navigating to the table and putting down the apple, which can be considered to be the two subgoals for executing the task. Each object interaction required to complete the task thus corresponds to a subgoal. Since every subgoal completion receives a positive reward, the number of subgoals completed at every time-step N_{sg} is known to the agent. The subgoals themselves are not known to the agent – we use attention on the text instruction to compute a latent subgoal representation.

7.4.1 Text subgoal inference

Given instruction τ composed of the tokens $(w_1, ..., w_n)$, we obtain the corresponding token embeddings $E = (e_1, ..., e_n)$ and use an RNN to encode the instruction to obtain a sequence

of contextualized token representations $H = (h_1, ..., h_n)$. We compute a *text subgoal* t_{sg} for a given time-step by computing attention on the instruction using N^e_{sg} as query where N^e_{sg} is a vector representation of N_{sg}. This is shown in Eq. (7.1) (Q, K, V are learnable parameter matrices).

$$\begin{aligned} t_{sg} &= \text{Attention}(\text{query} = N^e_{sg}, \text{keys} = \text{values} = H) \\ &= \sum_{h \in H} \frac{\exp((Qs)^\top (Kh))}{\sum_{h' \in H} \exp((Qs)^\top (Kh'))} Vh \end{aligned} \tag{7.1}$$

We expect the attention to focus on words in the instruction relevant to executing the current subgoal. For instance, if the agent is expected to interact with an apple, the attention module could learn to focus on the word 'apple'.

7.4.2 Cross-modal reasoning

Given the text subgoal t_{sg}, we use an attention mechanism to reason about objects in the scene within some distance to the agent. This helps the agent understand if objects of interest relevant to the subgoal are present nearby. Let the set of nearby scene objects be $O = \{o^1, ..., o^n\}$, where the $o^i \in \mathcal{I}$ are object ids provided by an oracle. The o^i's can thus be treated as indexes into an embedding table that produces object embeddings $O_e = \{o^1_e, ..., o^n_e\}$. The cross-modal attention is given by Eq. (7.2) where the text subgoal attends to the scene object embeddings (Q', K', V' are learnable parameter matrices). We augment the scene objects embeddings O_e with an additional learned embedding o^{no-obj}_e which is expected to absorb any probability mass not assigned to scene objects $O'_e = O_e \cup o^{no-obj}_e$. The attention produces an *object subgoal* embedding v_{sg}.

$$\begin{aligned} v_{sg} &= \text{Attention}(\text{query} = t_{sg}, \text{keys} = \text{values} = O'_e) \\ &= \sum_{o_e \in O'_e} \frac{\exp((Q' t_{sg})^\top (K' o_e))}{\sum_{o'_e \in O'_e} \exp((Q' t_{sg})^\top (K' o'_e))} V' o_e \end{aligned} \tag{7.2}$$

7.4.3 Policy learning

We use a deep Q-learning algorithm to train a policy (Mnih et al., 2013), where a neural network is trained to approximate the state-action value function $Q(s, a)$. Given the current observation, text subgoal and object subgoal, the state-action value for a navigation action $a \in \mathcal{A}_{\text{nav}}$ is given by Eq. (7.3), where f_{nav} is an MLP (multi-layer perceptron) and $e_{\text{obs}} = f_{\text{CNN}}(\text{obs})$ is a feature vector of the observation image computed using a CNN encoder.

$$Q_{\text{nav}}(s, a) = f_{\text{nav}}(a | e_{\text{obs}}, t_{sg}, v_{sg}) \tag{7.3}$$

Task type	Task description
`pick up pot`	Go to the stove and pick up the pot. Pick up the pot on the bottom right burner on the stove. Take the cooking pot from the stove.
`place spoon in pan`	get spoon on counter near salt shaker and put it away in pan near stove. Pick up the spoon from the table near the salt shaker and move it to the pan on the counter by the sink. Move spoon from the counter and into the pan.
`slice bread with knife`	Pick the knife and slice the bread. Take the knife with the yellow handle from the counter by the sink and use it to cut horizontal slices out of the loaf of bread on the white table. Pick up the sharp knife with a yellow handle, and slice the bread on the white table.

Table 7.1: Example task types and corresponding task descriptions. Note that the task descriptions are used for training and testing agents. The task types are not known to the agents.

The state-action values for interaction actions $a = (b, o) \in \mathcal{B} \times \mathcal{I}$ can be analogously modeled as in Eq. (7.4). We found it helpful to decompose the state-action value in an additive fashion over an action score f^a_{int} and an object score f^o_{int} as in Eq. (7.5). Intuitively, f^a_{int} learns to model action preferences, whereas f^o_{int} learns to ground text goals to physical objects. In addition to sharing parameters across actions and objects, this decomposition allows us to model state-action values of object interactions not experienced during training, as long as the specific interaction and the object were encountered. Unless specified otherwise we use the decomposed value function $Q^{\text{add}}_{\text{int}}$ in our experiments.

$$Q^{\text{full}}_{\text{int}}(s, a) = f_{\text{int}}(a|e_{\text{obs}}, t_{sg}, v_{sg}) \tag{7.4}$$

$$Q^{\text{add}}_{\text{int}}(s, a) = f^a_{\text{int}}(b|e_{\text{obs}}, t_{sg}, v_{sg}) + f^o_{\text{int}}(o|t_{sg}) \text{ where } a = (b, o) \in \mathcal{B} \times \mathcal{I} \tag{7.5}$$

In summary, the state-action value function is modeled as in Eq. (7.6).

$$Q(s, a) = \begin{cases} Q_{\text{nav}}(s, a); & a \in \mathcal{A}_{\text{nav}} \\ Q^{\text{add}}_{\text{int}}(s, a); & a \in \mathcal{A}_{\text{int}} \end{cases} \tag{7.6}$$

The overall model (see Fig. 7.2 for an illustration) including parameters of the subgoal inference (Eq. 7.1) and cross-modal reasoning (Eq. 7.2) components, as well as the MLPs in Eqs. (7.3) and (7.5) are trained end-to-end using a double-DQN algorithm (Van Hasselt et al., 2016). Once the model has been trained we construct a greedy policy by choosing actions with the highest state-action values for inference.

7.5 Experiments

7.5.1 Tasks

We use the AI2Thor (Kolve et al., 2017) environment as a testbed for our experiments. While there exist prior benchmarks that couple language instructions with embodied environments such as ALFRED Shridhar et al. (2020), they were not designed to study compositional generalization. We thus construct a new task setup that allows us to flexibly vary tasks and object arguments. We consider the following task types in our experiments,

- `pickup x`: Find and pick up object x
- `place x in y`: Find and pick up object x, followed by navigating to y and placing it.
- `slice x with y`: Secure cutting instrument y, find object x and slice it.

We use Amazon Mechanical Turk to collect natural language descriptions of tasks for training and evaluation. A turker is shown key observation frames during the execution of a particular task and is asked to describe in a sentence how they would describe the task to a robot. Turkers were instructed to do their best to correctly identify task relevant objects. But often descriptions from the turkers incorrectly identify objects such as identifying a potato as an avocado. Such descriptions were manually fixed so that the correct object identities are mentioned in the instructions. We collected 5 natural language descriptions each for 35 tasks that include `pickup`, `place` and `slice` tasks. The descriptions consist of 170 unique tokens and have an average length of 12 tokens. Table 7.1 shows example descriptions collected for some tasks. See appendix C.2 for instructions given to Turkers in the data collection process.

The pickup tasks are used for evaluating multi-task and zero-shot generalization with seen and unseen descriptions of tasks. We use 10 `pickup` tasks - `pickup X` where `X` $\in$ {`apple`, `bread`, `tomato`, `potato`, `lettuce`, `spoon`, `bread`, `butter knife`, `plate`, `pot`}. These tasks are used for evaluating generalization to seen and unseen descriptions of known short-horizon tasks. They are also used in generalization to longer horizon tasks as described later in this section.

The `place` and `slice` tasks are used for evaluating generalization to longer-horizon unseen tasks. Table 7.2 shows tasks used for training and evaluation. In addition to multitask generalization, we use these tasks to study zero-shot compositional generalization to unseen task descriptions. The unseen descriptions can correspond to tasks that were encountered during training, similar to the `pickup` tasks. A more challenging generalization scenario is to generalize to text descriptions of unseen tasks.

We consider two types of tasks in the latter scenario. The *obj-obj setting* examines the ability of the agent to generalize to tasks composed of unseen combinations of objects. For

	place tasks	**slice tasks**

Training tasks	`place apple in plate`
	`place butterknife in plate`
	`place spoon in plate`
	`place butterknife in pan`
	`place potato in pan`
	`place spoon in pan`
	`place apple in pot`
	`place butterknife in pot`
	`place potato in pot`
		`slice apple with knife`
		`slice tomato with knife`
		`slice bread with knife`
		`slice apple with butterknife`
		`slice potato with butterknife`
		`slice bread with butterknife`

Test tasks (obj-obj setting)	`place potato in plate`
	`place apple in pan`
	`place spoon in pot`
		`slice potato with knife`
		`slice tomato with butterknife`

Test tasks (task-obj setting)	`place knife in plate`
	`place knife in pan`
	`place knife in pot`
		`slice lettuce with knife`
		`slice lettuce with butterknife`

Table 7.2: Task types used for training and testing on place and slice tasks. The *obj-obj* setting considers test tasks composed of unseen combinations of objects. The *task-obj* setting considers generalization to unseen combinations of tasks and objects (e.g. learning to slice lettuce when taught how to slice objects and how to pickup lettuce).

instance, in the test task `place potato in plate`, the relevant objects `potato`, `plate` were encountered during training in tasks such as `place potato in pan` and `place apple in plate`.

The *task-obj setting* is a harder generalization problem where the agent is expected to generalize to unseen combinations of tasks and objects. For the test task `slice lettuce with knife`, the object `lettuce` was never observed in the context of a `slice` task during training. However, the agent has access to `pickup` tasks and is expected to learn to interact with lettuce by using the `pickup lettuce` task. This can be challenging because the agent was only taught how to pick up lettuce, and did not learn to associate lettuce with slice tasks.

The training tasks in Table 7.2 were designed in a way that each object argument appears in multiple tasks. Furthermore, when choosing object arguments for a given task type, we prioritized objects that appear in as many tasks as possible. For instance, in the pickup and place tasks setup, the objects were plate, pan, pot, spoon, etc. where each object appears in at least three of the training tasks. This ensures that there are enough occurrences of each object type for the agent to understand and ground the object type. It also helps the agent disentangle the notion of an object versus a task in a given text instruction.

7.5.2 Baselines and hyperparameters

Baselines We compare the proposed approach against the following baselines.

- **RNN** In this baseline we replace the attentional model with an RNN that produces an embedding of the text instruction. While this model can potentially work for unseen in-

structions, we examine if the encoding effectively captures the compositional information present in the instructions.

- **Gated Attention** This architecture (Chaplot et al., 2018) combines the instruction representation with the visual observation using a gated attention operation. The fused representation is fed to an MLP which models the state-action values. All models and baselines are trained using the DDQN Q-learning algorithm.

Hyperparameters Word embeddings and the RNN have representation size 32. Objects are represented by embeddings of size 32 from an embedding table. The CNN observation features have size 512 and the CNN encoder has 1.7M parameters, which constitues 90% of the overall model parameters. The MLPs in Eqs. (7.3) and (7.4) are single hidden layer MLPs with 256 hidden units and ReLU activation.

7.5.3 Results

Short-horizon tasks We first consider pickup tasks that involve a single object interaction. In these tasks the agent has to identify the object reference mentioned in the text description and then find and pick up the relevant object. We train and evaluate on 10 pickup task types. Four text descriptions of each task type are part of the training set and the remaining descriptions (i.e., 1 per task type) are part of the test set.

On the training and test descriptions, our agent trained from scratch achieves success rates of $0.9 \pm 0.3, 0.92 \pm 0.18$ respectively. Identifying the correct subgoal for these tasks involves paying attention to the verbs and nouns in the task description as well as the overall context. Table 7.3 visualizes the task attention in the subgoal inference component for a subset of test tasks, from which it is clear that the agent learns to focus on the relevant parts of the instruction.

pick up the silver butter knife closest to the edge of the counter .
the spoon is between the spatula and the fork on the left countertop ; pick it up .
move to the table , pick up the tomato .
pick up the lettuce from the table .
pick up the potato between the lettuce and the tomato .

Table 7.3: Visualizing task attention for pickup tasks. Words in darker shades received higher attention probabilities.

Longer-horizon Tasks We now consider tasks that involve two subgoals, which includes the place and slice tasks in Table 7.2. Jointly learning text grounding and subgoal inference for long horizon tasks can be challenging. We thus consider a curriculum learning strategy where an agent is gradually trained on tasks of increasingly longer horizon.

Tasks		Training tasks		Test tasks	
Descriptions		seen	unseen	unseen	unseen
				obj-obj	task-obj
Model	RNN	0.65 ± 0.50	0.65 ± 0.24	0.26 ± 0.30	0.13 ± 0.20
	Gated Attention	0.92 ± 0.20	0.85 ± 0.25	0.66 ± 0.30	0.34 ± 0.25
	Ours				
	(a) $Q_{\text{int}}^{\text{add}}$ (no cross modal)	0.89 ± 0.31	0.76 ± 0.35	0.84 ± 0.21	0.77 ± 0.30
	(b) $Q_{\text{int}}^{\text{full}}$ + cross modal	0.93 ± 0.26	0.85 ± 0.30	0.44 ± 0.34	0.34 ± 0.28
	(c) $Q_{\text{int}}^{\text{add}}$ + cross modal	$\mathbf{0.95 \pm 0.23}$	$\mathbf{0.87 \pm 0.27}$	$\mathbf{0.94 \pm 0.11}$	$\mathbf{0.91 \pm 0.23}$

Table 7.4: Task success rates (and standard deviation) of models under different generalization settings. Models are evaluated on seen/unseen descriptions of seen tasks and on unseen descriptions of unseen tasks. For unseen tasks, we further evaluate under unseen combinations of objects as well as unseen combinations of tasks and objects. Best numbers are boldfaced.

The agent is first pre-trained on the pickup tasks as described in the previous section, and then fine-tuned on the training tasks in Table 7.2.

Fig. 7.3 compares the learning progress of agents trained from scratch and an agent that has been pre-trained on the pickup tasks. The plot shows the success rate on training tasks during the course of training. The pre-trained agent learns twice as fast compared to the agent trained from scratch and achieves perfect success rate on training tasks.

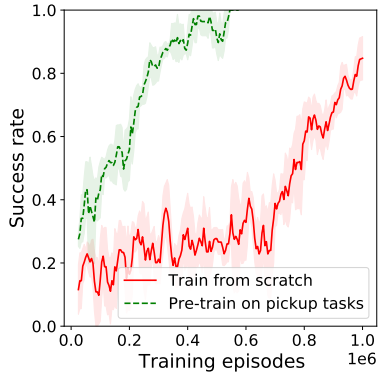


Figure 7.3: Learning progress of agent trained from scratch and pre-trained agent on place tasks.

Generalization Table 7.4 shows the average task completion success rate of models under different generalization scenarios. The RNN and Gated Attention baselines are limited by the fact that the text instruction is represented using the same encoding across all time-steps, which has limited ability to capture compositional information. The inductive bias of Gated Attention enables better performance, but it has difficulty generalizing to unseen tasks. The attention based model outperforms these baselines, which indicates that the attention mechanism helps exploit compositional information in the instruction better than a fixed encoding.

In addition to better performance, the attention model has the advantage of being more interpretable. Fig. 7.4 shows the agent's actions and the attention pattern over time for an example task. The agent learns to identify object references in the instruction and uses attention as a sub-goal representation. This mimics a hierarchical policy where a high-level controller provides a sub-goal and a low-level controller executes it Jiang et al. (2019b). The

Instruction attention Subgoal 1 bring the potato from the table to the plate on the right of the oven .
Subgoal 2 bring the potato from the table to the plate on the right of the oven .

Instruction attention Subgoal 1 pick up the butter knife on the counter , and horizontally slice the lettuce .
Subgoal 2 pick up the butter knife on the counter , and horizontally slice the lettuce .

Figure 7.4: Agent's observation at different time-steps while performing a place task and a slice task. The attention distribution in the text goal inference component while executing each subgoal is also given below the agent observations.

agent further learns to ground object references in the text instruction to objects in the scene. Notably, these attention patterns and grounding are learned from the reward signal alone without any other supervision. More example of agent trajectories are given in appendix C.1.

7.5.4 Ablations

We perform ablations to study the impact of cross-modal reasoning and decomposing the value function in an additive fashion.

Cross modal reasoning We examine model performance without the cross modal reasoning component. In this case the MLPs in Eqs. (7.3) and (7.5) only receive the text subgoal and observation encoding as inputs and the visual subgoal v_{sg} is omitted. From rows (a) and (c) in table Table 7.4 it is clear that the cross-modal reasoning components helps ground text in scene objects and enables better generalization across all settings.

Interaction Q-values We examine the benefit of decomposing the value function approximation of interaction actions in an additive fashion in $Q^{\text{add}}_{\text{int}}$ (Eq. (7.5)). We compare it against $Q^{\text{full}}_{\text{int}}$ (Eq. (7.4)), which treats each (interaction, object) pair as a separate atomic action.

Comparing rows (b), (c) in Table 7.4 we see that the additive decomposition is crucial for generalization to unseen tasks.

7.6 Conclusion

In this work we proposed attention based agents that can exploit the compositional nature of language instructions to generalize to unseen tasks. The policy mimics a hierarchical process where a text embedding obtained via attention represents the subgoal to be executed and the policy network executes the low level actions. The proposed method performs strongly against baselines on a testbed we created based on a photorealistic simulated environment and provides some interpretability.

Compared to existing benchmarks such as ALFRED we made simplifying assumptions such as oracle visual recognition, relatively short horizon tasks and generalization within single kitchen layout which allows us to focus on compositional generalization in embodied settings. However, the ideas presented here can potentially be combined with curriculum learning and learning from human demonstrations to perform complex tasks that require planning over hundreds of time-steps such as in the ALFRED setting, and we leave this to future work.

CHAPTER VIII

Conclusion and Future Work

Machine learning has seen incredible progress in the recent years. Despite this progress, machine learning models today are brittle and have fairly limited ability to generalize to new settings. They struggle to learn from limited supervision and often require large amounts of labelled training data.

This thesis addresses the limited supervision problem from two perspectives. First, I examine methods that exploit large amounts of unlabelled data to learn useful feature representations in a self-supervised manner. Such representations capture rich prior knowledge about the data, allowing them to be useful across many tasks, and enable data-efficient learning of new tasks. In the first part of this thesis (chapters III, IV), I presented methods to learn representations useful for a wide variety of text understanding tasks from unlabelled text. In particular, I show that contrastive learning can be effective to learn high quality sentence representations in an efficient manner compared to prior generative pre-training approaches.

In the second part of my thesis, I explore models and algorithms capable of learning from limited supervision. My work studies weakly supervised, few-shot and zero-shot learning settings with applications to text generation, sequence modeling, entity understanding and embodied control. I proposed methods to adapt transformer models to the few-shot regime (chapter V) and also explored how pre-trained transformer language models can be exploited to achieve strong zero-shot generalization capabilities (chapter VI). My work demonstrates that text descriptions are an effective means of building models that generalize to new domains and new tasks without needing to experience supervised data for the new domain/task. In particular, I considered a learning paradigm where models read and understand text in order to generalize to new domains and tasks and demonstrated these techniques for entity linking (chapter VI) and embodied control (chapter VII).

Below I discuss some interesting future directions building off of my work - The appli-

cability of my representation learning work to modern pre-trainig methods and leveraging text understanding models to build better AI.

Efficient self-supervised text representations with contrastive learning My Quick-Thoughts work shows that rich representations can be learned using contrastive training objectives in an efficient manner. Contrastive representation learning has further gained traction in other domains such as speech (Oord et al., 2018), vision (Chen et al., 2020) and RL (Srinivas et al., 2020). I believe that contrastive learning holds significant promise in building efficient algorithms for representation learning. The ideas in my work are complementary to, and can be combined with recent advances in language model based representations to learn constant-length text representations highly beneficial for fast retrieval applications. Further, recent unsupervised objectives for learning contextualized representations can also be cast under the contrastive learning framework (Kong et al., 2019). The contrastive learning framework thus offers considerable flexibility in the design of unsupervised representation learning algorithms while being a less expensive alternative to generative training.

Pre-training representations that can be easily adapted to new tasks My work on zero-shot Entity Linking is an instance of designing pre-training tasks with a certain target task or data domain in mind. Recent work have demonstrated that this approach is more widely applicable to other tasks and domains as well (Gururangan et al., 2019). I envision a more unified approach which avoids pre-training in multiple stages and instead exploits clusters of data that are naturally available in large unlabelled corpora to learn domain/task-aware representations. A mixture model of representations can then be easily adapted to new tasks by modulating the experts based on relevance/similarity to the target task or domain.

Acquiring strong generalization capabilities via text understanding Zeshel shows how Entity Linking systems can be made to generalize to unseen entities in unseen domains by forcing them to rely only on text descriptions of entities. More generally, I am excited about leveraging rich text information to expand the generalization capabilities of intelligent agents. I envision personal assistants that take commands in the form of text instructions and perform many practically useful tasks such as booking a flight ticket, reserving a restaurant or creating a calendar invite to meet someone.

I presented embodied agents that learn from text instructions in this thesis. Such sequential decision making problems are a good intermediate step towards the long term goal of realizing AI that acts intelligently based on text commands. They have many desiderata relevant to making progress on the long term goal - task requires reasoning across data from multiple modalities, sequential decision making is involved and there is a substantial gap between human performance and machine performance on these tasks.

It would be interesting to explore the use of pre-trained transformer models in this context and analyze whether the benefits seen in popular text understanding benchmarks such as GLUE are also applicable in this setting. Second, it is important to devise self-supervised objectives that enable learning of shared representations between text, images and actions. Finally, I would like to see if high level reasoning capabilities can be acquired by exploiting the compositional nature of text instructions such as generalizing to unseen combinations of skills acquired during training.

APPENDICES

APPENDIX A

Quick Thought vectors - Nearest neighbors

A.1 Nearest neighbors

Our model and the skip-thought model have conceptually similar objective functions. This suggests examining properties of the embedding spaces to better understand how they encode semantics. We consider a nearest neighbor retrieval experiment to compare the embedding spaces. We use a pool of 1M sentences from a Wikipedia dump for this experiment. For a given query sentence, the best neighbor determined by cosine distance in the embedding space is retrieved.

Table A.1 shows a random sample of query sentences from the dataset and the corresponding retrieved sentences. These examples show that our retrievals are often more related to the query sentence compared to the skip-thought model. It is interesting to see in the first example that the model identifies a sentence with similar meaning even though the main clause and conditional clause are in a different order. This is in line with our goal of learning representations that are less sensitive to the form in which meaning is expressed.

Query	Seizures may occur as the glucose falls further .
ST	It may also occur during an excessively rapid entry into autorotation .
QT	When brain glucose levels are sufficiently low , seizures may result .
Query	This evidence was only made public after both enquiries were completed .
ST	This visa was provided for under Republic Act No .
QT	These evidence were made public by the United States but concealed the names of sources .
Query	He kept both medals in a biscuit tin .
ST	He kept wicket for Middlesex in two first-class cricket matches during the 1891 County Championship .
QT	He won a three medals at four Winter Olympics .
Query	The American alligator is the only known natural predator of the panther .
ST	Their mascot is the panther .
QT	The American alligator is a fairly large species of crocodilian .
Query	Several of them died prematurely : Carmen and Toms very young , while Carlos and Pablo both died .
ST	At the age of 13 , Ahmed Sher died .
QT	Many of them died in prison .
Query	Music for " Expo 2068 " originated from the same studio session .
ST	His 1994 work " Dialogue " was premiered at the Merkin Concert Hall in New York City .
QT	Music from " Korra " and " Avatar " was also played in concert at the PlayFest festival in Mlaga , Spain in September 2014 .
Query	Mohammad Ali Jinnah yielded to the demands of refugees from the Indian states of Bihar and Uttar Pradesh , who insisted that Urdu be Pakistan 's official language .
ST	Georges Charachidz , a historian and linguist of Georgian origin under Dumzil 's tutelage , became a noted specialist of the Caucasian cultures and aided Dumzil in the reconstruction of the Ubykh language .
QT	Wali Mohammed Wali 's visit thus stimulated the growth and development of Urdu Ghazal in Delhi .
Query	The PCC , together with the retrosplenial cortex , forms the retrosplenial gyrus .
ST	The Macro domain from human , macroH2A1.1 , binds an NAD metabolite O-acetyl-ADP-ribose .
QT	The PCC forms a part of the posteromedial cortex , along with the retrosplenial cortex (Brodmann areas 29 and 30) and precuneus (located posterior and superior to the PCC) .

Table A.1: Nearest neighbors retrieved by the skip-thought model (ST) and our model (QT).

APPENDIX B

Zero-shot Entity Linking - Model predictions and errors

B.1 Examining model errors and predictions

In tables B.1, B.2, B.3, B.4 we show some example mentions and model predictions. For each instance, the examples show the correct gold entity and the top-5 predictions from the model. Examples show 32 token contexts centered around mentions and the first 32 tokens of candidate entity documents.

Coronation Street		
Mention	Robbie pulled over the ambulance with a van and used a gun to get the **Prison Officer** with Tony to release him . He integrated himself with the Street residents , finding	
Gold Entity	Prison Officer (Episode 7351)	The unnamed Prison Officer was on duty during May 2010 in the Highfield Prison dining room when Tony Gordon provoked a fight with a fellow inmate
	Top-5 predictions	
✓	Prison Officer (Episode 7351)	The unnamed Prison Officer was on duty during May 2010 in the Highfield Prison dining room when Tony Gordon provoked a fight with a fellow inmate
	Inmate (Episode 7351)	The Inmate was an unnamed fellow prisoner of Tony Gordon in Highfield Prison . Tony provoked a fight in the dining room with the inmate by staring
	Police Officer (Simon Willmont)	The unnamed Police Officer was on duty at Weatherfield Police Station in March 2010 when Peter Barlow was released from custody following his arrest as he
	Prison Officer (Bill Armstrong)	The Prison Officer looked after the incarceration of three Coronation Street residents : In November 2000 he was on duty at Strangeways Jail when Jim McDonald
	Robbie Sloane	Quietly spoken Robbie Sloane was Tony Gordon ' s henchman and a convicted murderer , who he met while sharing a cell at Highfield Prison in 2010 . When Robbie

Table B.1: Mention and entity candidates from Coronation Street.

Muppets		
Mention	Bean Bunny was introduced during the seventh season of " Muppet Babies " , and a **pre - teen Bean** would later be featured as part of the Muppet Kids series . Bean was active	
Gold Entity	Bean Bunny (Muppet Kids)	A young version of Bean Bunny made a few appearances in the Muppet Kids books and video games . Young Bean moves to the Muppet Kids
	Top-5 predictions	
	Baby Bean Bunny	Baby Bean Bunny appeared in the late 1989 / 1990 seasons of " Muppet Babies " as a baby version of Bean Bunny . He joined the other babies
✓	Bean Bunny (Muppet Kids)	A young version of Bean Bunny made a few appearances in the Muppet Kids books and video games . Young Bean moves to the Muppet Kids
	Bean Bunny	Bean Bunny first appeared in 1986 as the star of the TV special " The Tale of the Bunny Picnic " . The cute bunny was part of a family
	Piggy (Muppet Kids)	A pre - teen version of Miss Piggy , as seen in the " Muppet Kids " books and video games . Piggy lives in a fancy
	Muppet Kids	Muppet Kids was a series of books and educational software made in the 1990s , featuring young , pre - teen versions of the principal franchise characters . Characters included

Table B.2: Mention and entity candidates from Muppets.

	Ice Hockey	
Mention	1979 - 80 PCJHL Season This is a list of **Peace - Cariboo Junior Hockey League** Standings for the 1979 - 80 season . This was the PCJHL ' s final	
Gold Entity	Rocky Mountain Junior Hockey League	The Rocky Mountain Junior Hockey League was a Canadian Junior " A " ice hockey league in British Columbia . History . Promoted to"
	Top-5 predictions	
	Peace Junior Hockey League	Hockey League Peace Junior Hockey League is a League that started in the 1960 ' s and ended in 1975 . Then change its name to Peace Cariboo junior Hockey
	Cariboo Hockey League	The Cariboo Hockey League was a Senior and Intermediate hockey league in the Cariboo District of British Columbia , Canada . History . The league began in the 1955
	Cariboo Junior League	The Cariboo Junior League operated in northern British Columbia in the 1963 - 64 season . Its champion was eligible for the British Columbia Junior Playoffs . The league
✓	Rocky Mountain Junior Hockey League	The Rocky Mountain Junior Hockey League was a Canadian Junior " A " ice hockey league in British Columbia . History . Promoted to"
	North West Junior Hockey League	The North West Junior Hockey League is a Junior " B " ice hockey league operating in the Peace River region of Alberta and British Columbia ,

Table B.3: Mention and entity candidates from Ice Hockey.

	Elder Scrolls	
Mention	to get everyone to safety . Rolunda ' s brother is one of those people . **The Frozen Man** . Rolunda ' s brother Eiman has ventured into Orkey ' s Hollow to find	
Gold Entity	The Frozen Man (Quest)	The Frozen Man is a quest available in The Elder Scrolls Online. It involves finding a Nord who has been trapped in ice by a mysterious " Frozen Man
	Top-5 predictions	
✓	The Frozen Man (Quest)	The Frozen Man is a quest available in The Elder Scrolls Online. It involves finding a Nord who has been trapped in ice by a mysterious " Frozen Man
	The Frozen Man	The Frozen Man is an insane Bosmer ghost found in Orkey ' s Hollow . He says he was in a group of people inside the cave when it
	Kewan	Kewan is a Redguard worshipper of the Daedric Prince Peryite . He is frozen in a trance that relates to the Daedric quest , but can be unfrozen in completion the
	Stromgruf the Steady	Stromgruf the Steady is the Nord who is found in the Grazelands of Vvardenfell , west of Pulk and east of Vassamsi Grotto (Online) . He is
	Maren the Seal	Maren the Seal is a Nord hunter and worshipper of the Daedric Prince Peryite . She is frozen in a trance that relates to the Daedric Prince ' s

Table B.4: Mention and entity candidates from Elder Scrolls.

APPENDIX C

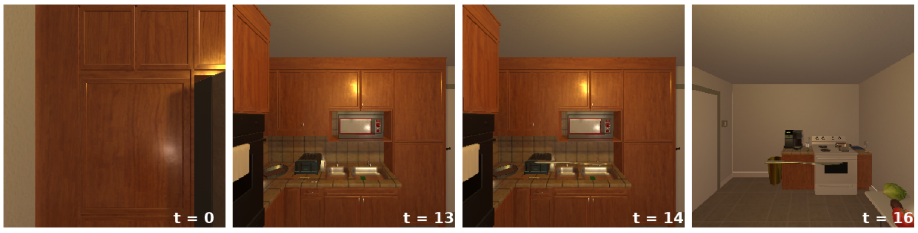
Compositional task generalization

C.1 Sample agent trajectories

Subgoal 1 bring the apple from the counter to the pan on the counter by the microwave .
Subgoal 2 bring the apple from the counter to the pan on the counter by the microwave .

Subgoal 1 take the knife from the counter by the trash can and put it on the plate on the counter to the right of the oven .
Subgoal 2 take the knife from the counter by the trash can and put it on the plate on the counter to the right of the oven .

Subgoal 1 pick up the yellow knife on the counter in the kitchen and cut the potato that is located on the white table
Subgoal 2 pick up the yellow knife on the counter in the kitchen and cut the potato that is located on the white table



Subgoal 1 pick up the silver butter knife from the counter by the sink and bring it to the white table to slice lettuce into thin horizontal slices .
Subgoal 2 pick up the silver butter knife from the counter by the sink and bring it to the white table to slice lettuce into thin horizontal slices .

Figure C.1: Agent's observation at different time-steps while performing place and slice tasks. The attention distribution in the text goal inference component while executing each subgoal is also given below the agent observations.

C.2 Collecting task descriptions from Mechanical Turk

Fig. C.2 shows an example image and instructions shown to Mechanical Turkers to collect natural language task descriptions.

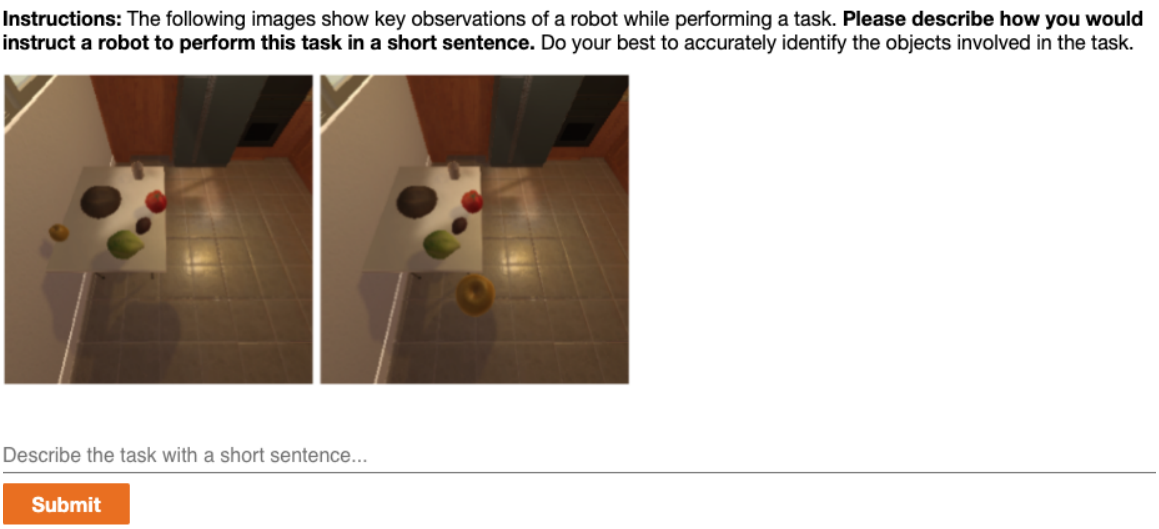


Figure C.2: Example of a HIT (Human Intelligence Task) shown to Turkers in Amazon Mechanical Turk.

BIBLIOGRAPHY

BIBLIOGRAPHY

Adi, Y., Kermany, E., Belinkov, Y., Lavi, O., and Goldberg, Y. (2017). Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *ICLR*.

Akata, Z., Reed, S., Walter, D., Lee, H., and Schiele, B. (2015). Evaluation of output embeddings for fine-grained image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2927–2936.

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., and Van Den Hengel, A. (2018). Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683.

Andreas, J. (2019). Good-enough compositional data augmentation. *arXiv preprint arXiv:1904.09545*.

Andreas, J., Klein, D., and Levine, S. (2017). Modular Multitask Reinforcement Learning with Policy Sketches. *arXiv:1611.01796 [cs]*. arXiv: 1611.01796.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and De Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*, pages 3981–3989.

Arora, S., Liang, Y., and Ma, T. (2016). A simple but tough-to-beat baseline for sentence embeddings.(2016).

Ba, J., Hinton, G. E., Mnih, V., Leibo, J. Z., and Ionescu, C. (2016). Using fast weights to attend to the recent past. In *Advances in Neural Information Processing Systems*, pages 4331–4339.

Bao, Y., Wu, M., Chang, S., and Barzilay, R. (2019). Few-shot text classification with distributional signatures. *arXiv preprint arXiv:1908.06039*.

Barrault, L., Bojar, O., Costa-jussà, M. R., Federmann, C., Fishel, M., Graham, Y., Haddow, B., Huck, M., Koehn, P., Malmasi, S., Monz, C., Müller, M., Pal, S., Post, M., and Zampieri, M. (2019). Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy. Association for Computational Linguistics.

Barzilay, R. and Elhadad, N. (2002). Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research*, pages 35–55.

Barzilay, R. and Lapata, M. (2008). Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Barzilay, R. and Lee, L. (2004). Catching the drift: Probabilistic content models, with applications to generation and summarization. *arXiv preprint cs/0405039*.

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., et al. (2016). Deepmind lab. *arXiv preprint arXiv:1612.03801*.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2015). Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Branavan, S., Silver, D., and Barzilay, R. (2012). Learning to win by reading manuals in a monte-carlo framework. *Journal of Artificial Intelligence Research*, 43:661–704.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Bunescu, R. and Pasca, M. (2006). Using encyclopedic knowledge for named entity disambiguation. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Burstein, J., Tetreault, J., and Andreyev, S. (2010). Using entity-based features to model coherence in student essays. In *Human language technologies: The 2010 annual conference of the North American chapter of the Association for Computational Linguistics*, pages 681–684. Association for Computational Linguistics.

Carvalho, W., Liang, A., Lee, K., Sohn, S., Lee, H., Lewis, R. L., and Singh, S. (2020). Reinforcement learning for sparse-reward object-interaction tasks in first-person simulated 3d environments. *arXiv preprint arXiv:2010.15195*.

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Céspedes, M., Yuan, S., Tar, C., et al. (2018). Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Chaplot, D. S. and Salakhutdinov, R. (2018). Knowledge-based word sense disambiguation using topic models. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Chaplot, D. S., Sathyendra, K. M., Pasumarthi, R. K., Rajagopal, D., and Salakhutdinov, R. (2018). Gated-attention architectures for task-oriented language grounding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Chen, M., Xu, Z., Weinberger, K., and Sha, F. (2012). Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning*.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*.

Chen, X., Qiu, X., and Huang, X. (2016). Neural sentence ordering. *arXiv preprint arXiv:1607.06952*.

Cheng, J. and Lapata, M. (2016). Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.

Chomsky, N. (2009). *Syntactic structures*. De Gruyter Mouton.

Chung, J., Gül10031cehre, C., Cho, K., and Bengio, Y. (2015). Gated feedback recurrent neural networks. In *ICML*, pages 2067–2075.

Cohen, D., Mitra, B., Hofmann, K., and Croft, W. B. (2018). Cross domain regularization for neural ranking models using adversarial learning. In *The 41st International ACM SIGIR Conference on Research; Development in Information Retrieval*.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Dai, A. M. and Le, Q. V. (2015). Semi-supervised sequence learning. *arXiv preprint arXiv:1511.01432*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430.

Dolan, B., Quirk, C., and Brockett, C. (2004). Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics.

Eisenstein, J. (2018). *Natural Language Processing*. MIT Press.

Elhoseiny, M., Saleh, B., and Elgammal, A. (2013). Write a classifier: Zero-shot learning using purely textual descriptions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2584–2591.

Elsner, M., Austerweil, J. L., and Charniak, E. (2007). A unified local and global model for discourse coherence. In *HLT-NAACL*, pages 436–443.

Fei-Fei, L., Fergus, R., and Perona, P. (2006). One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4).

Fink, M. (2005). Object classification from a single example utilizing class relevance metrics. In *In Advances in Neural Information Processing Systems*, pages 449—456.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org.

Gan, Z., Pu, Y., Henao, R., Li, C., He, X., and Carin, L. (2016). Unsupervised learning of sentence representations using convolutional neural networks. *arXiv preprint arXiv:1611.07897*.

Ganea, O.-E. and Hofmann, T. (2017). Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Ghiasi, G., Lee, H., Kudlur, M., Dumoulin, V., and Shlens, J. (2017). Exploring the structure of a real-time, arbitrary neural artistic stylization network. *arXiv preprint arXiv:1705.06830*.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning*.

Grosz, B. J., Weinstein, S., and Joshi, A. K. (1995). Centering: A framework for modeling the local coherence of discourse. *Computational linguistics*, 21(2):203–225.

Guinaudeau, C. and Strube, M. (2013). Graph-based local coherence modeling. In *ACL (1)*, pages 93–103.

Guo, J., Shah, D., and Barzilay, R. (2018). Multi-source domain adaptation with mixture of experts. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Gupta, N., Singh, S., and Roth, D. (2017a). Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Gupta, S., Davidson, J., Levine, S., Sukthankar, R., and Malik, J. (2017b). Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625.

Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., and Smith, N. A. (2019). Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Han, L., Kashyap, A. L., Finin, T., Mayfield, J., and Weese, J. (2013). UMBC_EBIQUITY-CORE: Semantic Textual Similarity Systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics.

He, L., Lee, K., Levy, O., and Zettlemoyer, L. (2018). Jointly predicting predicates and arguments in neural semantic role labeling. *arXiv preprint arXiv:1805.04787*.

Hermann, K. M. and Blunsom, P. (2013). Multilingual distributed representations without word alignment. *arXiv preprint arXiv:1312.6173*.

Hill, F., Cho, K., and Korhonen, A. (2016). Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483*.

Hill, F., Cho, K., Korhonen, A., and Bengio, Y. (2015). Learning to understand phrases by embedding the dictionary. *arXiv preprint arXiv:1504.00548*.

Hill, F., Lampinen, A., Schneider, R., Clark, S., Botvinick, M., McClelland, J. L., and Santoro, A. (2019). Environmental drivers of systematicity and generalization in a situated agent. *arXiv preprint arXiv:1910.00571*.

Hinton, G. E. and Plaut, D. C. (1987). Using fast weights to deblur old memories. In *Proceedings of the ninth annual conference of the Cognitive Science Society*, pages 177–186.

Hochreiter, S., Younger, A. S., and Conwell, P. R. (2001). Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pages 87–94. Springer.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for nlp.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2014). On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*.

Jernite, Y., Bowman, S. R., and Sontag, D. (2017). Discourse-based objectives for fast unsupervised sentence representation learning. *arXiv preprint arXiv:1705.00557*.

Ji, Y. and Eisenstein, J. (2013). Discriminative improvements to distributional sentence similarity. In *EMNLP*, pages 891–896.

Jiang, Y., Gu, S., Murphy, K., and Finn, C. (2019a). Language as an abstraction for hierarchical deep reinforcement learning. *arXiv preprint arXiv:1906.07343*.

Jiang, Y., Gu, S. S., Murphy, K. P., and Finn, C. (2019b). Language as an Abstraction for Hierarchical Deep Reinforcement Learning. In Wallach, H., Larochelle, H., Beygelzimer, A., AlchÃ©-Buc, F. d., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 9419–9431. Curran Associates, Inc.

Johnson, J., Hariharan, B., Van Der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. (2017). Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910.

Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., and Levy, O. (2020). Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137.

Kenter, T., Borisov, A., and de Rijke, M. (2016). Siamese cbow: Optimizing word embeddings for sentence representations. *arXiv preprint arXiv:1606.04640*.

Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., and Socher, R. (2019). Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Kiddon, C., Zettlemoyer, L., and Choi, Y. (2016). Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. In *Advances in neural information processing systems*.

Klein, B., Lev, G., Sadeh, G., and Wolf, L. (2015). Associating neural word embeddings with deep image representations using fisher vectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4437–4446.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Gordon, D., Zhu, Y., Gupta, A., and Farhadi, A. (2017). AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*.

Kong, L., d'Autume, C. d. M., Ling, W., Yu, L., Dai, Z., and Yogatama, D. (2019). A mutual information maximization perspective of language representation learning. In *International Conference on Learning Representations*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kumar, A. and Daume III, H. (2012). Learning task grouping and overlap in multi-task learning. In *International Conference on Machine Learning*.

Lake, B. and Baroni, M. (2018). Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2873–2882. PMLR.

Lake, B. M. (2019). Compositional generalization through meta sequence-to-sequence learning. *arXiv preprint arXiv:1906.05381*.

Lample, G., Subramanian, S., Smith, E., Denoyer, L., Ranzato, M., and Boureau, Y.-L. (2019). Multiple-attribute text rewriting. In *International Conference on Learning Representations*.

Lapata, M. (2003). Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 545–552. Association for Computational Linguistics.

Lapata, M. (2006). Automatic evaluation of information ordering: Kendall's tau. *Computational Linguistics*, 32(4):471–484.

Larsen, A. B. L., Sønderby, S. K., Larochelle, H., and Winther, O. (2015). Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*.

Le, P. and Titov, I. (2018). Improving entity linking by modeling latent relations between mentions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Li, J., Chen, X., Hovy, E., and Jurafsky, D. (2015a). Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*.

Li, J. and Hovy, E. H. (2014). A model of coherence based on distributed sentence representation. In *EMNLP*, pages 2039–2048.

Li, J. and Jurafsky, D. (2016). Neural net models for open-domain discourse coherence. *arXiv preprint arXiv:1606.01545*.

Li, J., Luong, M.-T., and Jurafsky, D. (2015b). A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Lichman, M. (2013). UCI machine learning repository.

Lin, R., Liu, S., Yang, M., Li, M., Zhou, M., and Li, S. (2015). Hierarchical recurrent neural network for document modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 899–907.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Ling, X., Singh, S., and Weld, D. S. (2015). Design challenges for entity linking. *Transactions of the Association for Computational Linguistics*.

Liu, X., He, P., Chen, W., and Gao, J. (2019). Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Logeswaran, L., Chang, M.-W., Lee, K., Toutanova, K., Devlin, J., and Lee, H. (2019). Zero-shot entity linking by reading entity descriptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Logeswaran, L. and Lee, H. (2018). An efficient framework for learning sentence representations. *arXiv preprint arXiv:1803.02893*.

Logeswaran, L., Lee, H., and Radev, D. (2018). Sentence ordering and coherence modeling using recurrent neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Louis, A. and Nenkova, A. (2012). A coherence model based on syntactic patterns. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1157–1168. Association for Computational Linguistics.

Lynch, C. and Sermanet, P. (2020). Grounding Language in Play. *arXiv:2005.07648 [cs]*. arXiv: 2005.07648.

Mao, J., Xu, W., Yang, Y., Wang, J., Huang, Z., and Yuille, A. (2014). Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*.

Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R., and Zamparelli, R. (2014). A sick cure for the evaluation of compositional distributional semantic models. In *LREC*, pages 216–223.

Maurer, A., Pontil, M., and Romera-Paredes, B. (2013). Sparse coding for multitask and transfer learning. In *International conference on machine learning*, pages 343–351.

McCann, B., Bradbury, J., Xiong, C., and Socher, R. (2017). Learned in translation: Contextualized word vectors. *arXiv preprint arXiv:1708.00107*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*.

Milne, D. and Witten, I. H. (2008). Learning to link with wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*.

Miltsakaki, E. and Kukich, K. (2004). Evaluation of text coherence for electronic essay scoring systems. *Natural Language Engineering*, 10(01):25–55.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. (2018). A simple neural attentive meta-learner. In *International Conference on Learning Representations*.

Mnih, A. and Hinton, G. E. (2009). A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Murty, S., Verga, P., Vilnis, L., Radovanovic, I., and McCallum, A. (2018). Hierarchical losses and new resources for fine-grained entity typing and linking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Nallapati, R., Zhou, B., and Ma, M. (2016). Classify or select: Neural architectures for extractive document summarization. *arXiv preprint arXiv:1611.04244*.

Nguyen, D. T. and Joty, S. (2017). A neural local coherence model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1320–1330.

Nichol, A. and Schulman, J. (2018). Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2.

Noroozi, M. and Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer.

Norouzi, M., Mikolov, T., Bengio, S., Singer, Y., Shlens, J., Frome, A., Corrado, G. S., and Dean, J. (2013). Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650*.

Oh, J., Singh, S., Lee, H., and Kohli, P. (2017). Zero-Shot Task Generalization with Multi-Task Deep Reinforcement Learning. *arXiv:1706.05064 [cs]*. arXiv: 1706.05064.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Pang, B. and Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Pang, B. and Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

Parisotto, E., Song, F., Rae, J., Pascanu, R., Gulcehre, C., Jayakumar, S., Jaderberg, M., Kaufman, R. L., Clark, A., Noury, S., Botvinick, M., Heess, N., and Hadsell, R. (2019). Stabilizing transformers for reinforcement learning. *arXiv:1910.06764*.

Park, C. C. and Kim, G. (2015). Expressing an image stream with a sequence of natural sentences. In *Advances in Neural Information Processing Systems*, pages 73–81.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. *arXiv preprint arXiv:1705.05363*.

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. (2016). Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. (2018). Film: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018a). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018b). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Purushwalkam, S., Nickel, M., Gupta, A., and Ranzato, M. (2019). Task-driven modular networks for zero-shot compositional learning. *arXiv preprint arXiv:1905.05908*.

Radev, D. R., Joseph, M. T., Gibson, B., and Muthukrishnan, P. (2009). A Bibliometric and Network Analysis of the field of Computational Linguistics. *Journal of the American Society for Information Science and Technology*.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding with unsupervised learning. Technical report, OpenAI.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Ravi, S. and Larochelle, H. (2016). Optimization as a model for few-shot learning.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Rethmeier, N. and Augenstein, I. (2021). A primer on contrastive pretraining in language processing: Methods, lessons learned and perspectives. *arXiv preprint arXiv:2102.12982*.

Romera-Paredes, B. and Torr, P. (2015). An embarrassingly simple approach to zero-shot learning. In *International conference on machine learning*, pages 2152–2161. PMLR.

Roth, D., Ji, H., Chang, M.-W., and Cassidy, T. (2014). Wikification and beyond: The challenges of entity and concept grounding. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: Tutorials*.

Ruis, L., Andreas, J., Baroni, M., Bouchacourt, D., and Lake, B. M. (2020). A Benchmark for Systematic Generalization in Grounded Language Understanding. *arXiv:2003.05161 [cs]*. arXiv: 2003.05161.

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. (2019). Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. (2016). One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065*.

Schmidhuber, J. (1987). *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München.

Schmidhuber, J. (1992). Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139.

Sennrich, R., Haddow, B., and Birch, A. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Shen, T., Ott, M., Auli, M., and Ranzato, M. (2019). Mixture models for diverse machine translation: Tricks of the trade. In *International Conference on Machine Learning*.

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. (2020). ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. *arXiv:1912.01734 [cs]*. arXiv: 1912.01734.

Sil, A., Cronin, E., Nie, P., Yang, Y., Popescu, A.-M., and Yates, A. (2012). Linking named entities to any database. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Sil, A., Ji, H., Roth, D., and Cucerzan, S.-P. (2018). Multi-lingual entity discovery and linking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Tutorial Abstracts*.

Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087.

Socher, R., Ganjoo, M., Sridhar, H., Bastani, O., Manning, C. D., and Ng, A. Y. (2013a). Zero-shot learning through cross-modal transfer. *arXiv preprint arXiv:1301.3666*.

Socher, R., Huang, E. H., Pennington, J., Ng, A. Y., and Manning, C. D. (2011). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*, volume 24, pages 801–809.

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., Potts, C., et al. (2013b). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y. (2019). Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*.

Soricut, R. and Marcu, D. (2006). Discourse generation using utility-trained coherence models. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 803–810. Association for Computational Linguistics.

Srinivas, A., Laskin, M., and Abbeel, P. (2020). Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*.

Stickland, A. C. and Murray, I. (2019). Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *International Conference on Machine Learning*.

Subramanian, S., Trischler, A., Bengio, Y., and Pal, C. J. (2018). Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017a). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017b). Attention is all you need. In *Advances in Neural Information Processing Systems*.

Vendrov, I., Kiros, R., Fidler, S., and Urtasun, R. (2015). Order-embeddings of images and language. *arXiv preprint arXiv:1511.06361*.

Vinyals, O., Bengio, S., and Kudlur, M. (2015a). Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. (2016). Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015b). Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2674–2682.

Voorhees, E. M. and Buckland, L. (2003). Overview of the trec 2003 question answering track. In *TREC*, volume 2003, pages 54–68.

Wang, H., Zheng, J. G., Ma, X., Fox, P., and Ji, H. (2015). Language and domain independent entity linking with quantified collective validation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Wang, X. and Gupta, A. (2015). Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802.

Wiebe, J., Wilson, T., and Cardie, C. (2005). Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2):165–210.

Wieting, J., Bansal, M., Gimpel, K., and Livescu, K. (2015). Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

Wieting, J. and Gimpel, K. (2017). Revisiting recurrent networks for paraphrastic sentence embeddings. *arXiv preprint arXiv:1705.00364*.

Wieting, J., Mallinson, J., and Gimpel, K. (2017). Learning paraphrastic sentence embeddings from back-translated bitext. *arXiv preprint arXiv:1706.01847*.

Wu, L., Petroni, F., Josifoski, M., Riedel, S., and Zettlemoyer, L. (2019). Scalable zero-shot entity linking with dense entity retrieval. *arXiv preprint arXiv:1911.03814*.

Yang, Y. and Eisenstein, J. (2015). Unsupervised multi-domain adaptation with feature embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Zellers, R., Holtzman, A., Rashkin, H., Bisk, Y., Farhadi, A., Roesner, F., and Choi, Y. (2019). Defending against neural fake news. In *Neural Information Processing Systems*.

Zhao, H., Lu, Z., and Poupart, P. (2015). Self-adaptive hierarchical sentence model. In *IJCAI*, pages 4069–4076.

Zhao, H., Zhang, S., Wu, G., Moura, J. M., Costeira, J. P., and Gordon, G. J. (2018). Adversarial multiple source domain adaptation. In *Advances in Neural Information Processing Systems*.

Zhu, Y., Gordon, D., Kolve, E., Fox, D., Fei-Fei, L., Gupta, A., Mottaghi, R., and Farhadi, A. (2017). Visual semantic planning using deep successor representations. In *Proceedings of the IEEE international conference on computer vision*, pages 483–492.

Zintgraf, L. M., Shiarlis, K., Kurin, V., Hofmann, K., and Whiteson, S. (2019). Fast context adaptation via meta-learning. In *International Conference on Machine Learning*.